НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО

Факультет інформатики та обчислювальної техніки
(назва факультету, інституту)

Кафедра автоматизованих систем обробки інформації і управління
(назва кафедри)

"На правах рукопису"
УДК 004.932.2

«До захисту допущено»
Завідувач кафедри

_____ О.А.Павлов
(підпис) (ініціали, прізвище)

“____” _____ 20*18* р.

МАГІСТЕРСЬКА ДИСЕРТАЦІЯ

на здобуття ступеня магістра

за спеціальністю *122 Комп'ютерні науки та інформаційні технології*
(код та назва спеціальності)

спеціалізацією *Інформаційні управляючі системи та технології*
(код та назва спеціалізації)

на тему: Паралельний адаптивний вирішувач для лінійних систем на основі *нейронної мережі*

Виконав: студент *VI* курсу групи ІС-63м
(шифр групи)

Душутін Владислав Володимирович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник *Проф. д.т.н., проф. Хіміч О.М.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант *к.т.н., доц. Жданова О.Г.* _____
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018

РЕФЕРАТ

Магістерська дисертація: 100 с., 15 рис., 14 табл., 1 додаток, 83 джерела.

Зараз одним з основних етапів при дослідженні об’єктів, явищ і процесів різної природи є математичне моделювання і пов’язаний ним комп’ютерний експеримент. Чисельні експерименти дають можливість, як планувати натурний експеримент, так і отримувати нові знання про ті процеси і явища для яких утруднений, або взагалі неможливий натурний експеримент. Велика кількість математичних моделей після виконання відповідних перетворень можуть бути описанні системами лінійних алгебраїчних рівнянь (СЛАР) з розрідженими матрицями.

Основною особливістю таких систем є їхні великі порядки і невелика кількість ненульових елементів. Великі порядки СЛАР виникають за рахунок того, що дослідники хочуть отримати якомога достовірніші результати, через це будуються більш деталізовані моделі. Мала кількість ненульових елементів пояснюється особливостями дискретизації моделі. Зокрема, системи рівнянь з розрідженими матрицями виникають у задачах аналізу міцності конструкцій у цивільному та промисловому будівництві, фільтрації, тепло- та масо переносу, тощо. Область застосування методів розв’язування СЛАР з розрідженими матрицями постійно розширюється. Через це виникає інтерес до проблеми побудови ефективних методів розв’язання таких систем, порядки яких перевищую сотні тисяч.

Класичні результати, що стосуються розробки методів розв’язання СЛАР з розрідженими матрицями висвітлюються у ряді монографій американських і вітчизняних авторів: А. Джорджа, Дж. Лю, С. Писанецьки, Дж. Голуба, Р. Тюарсона, І.А. Блатова, М.Е. Ексаревської та інших.

Також зростають вимоги до обчислювальної техніки, що використовується для проведення комп’ютерного експерименту. Вона повинна забезпечувати достатню швидкодію і мати необхідну кількість ресурсів, щоб результат експерименту можна було отримати за досить невеликий проміжок часу. Зараз на ринку представлені багато різних архітектур комп’ютерів з

паралельною організацією обчислень. Найбільш продуктивними є платформи так званої «гібридної» архітектури. Дані системи поєднують у собі MIMD- (multiple instructions – multiple data) та SIMD-архітектури (single instruction – multiple data), а саме у системі з багатоядерними процесорами обчислення прискорюються за рахунок графічного прискорювача. Отже одним з ефективних підходів до розв'язання СЛАР з розрідженими матрицями є побудова паралельних алгоритмів, що враховують особливості архітектури комп'ютера.

Основними проблемами розробки ефективних паралельних алгоритмів є: аналіз структури матриці, або приведення її до відповідного вигляду, застосовуючи відповідні алгоритми перетворення; вибір ефективної декомпозиції даних; визначення ефективної кількості процесорних ядер і графічних прискорювачів, що використовуються для обчислень; визначення топології міжпроцесних зв'язків, яка зменшує кількість комунікацій і синхронізацій.

Саме для аналізу структури розрідженої матриці використовується нейрона мережа, яка дозволить виділити групи ненульових елементів, які можуть оброблятись незалежно. За результатами аналізу буде будуватись декомпозиція даних та обиратись кількість обчислювальних ядер, що забезпечить найкоротший час розрахунків для конкретної структури матриці.

Мета та завдання дослідження. Метою роботи є розробка та дослідження паралельних методів та комп'ютерних алгоритмів для дослідження та розв'язування СЛАР з розрідженими матрицями нерегулярної структури на комп'ютерах MIMD-архітектури та комбінації MIMD- і SIMD-архітектури, апробація алгоритмів при математичному моделюванні у прикладних задачах.

До завдань дослідження належать:

- розробка та дослідження ітераційних паралельних алгоритмів для СЛАР з розрідженими матрицями нерегулярної структури з наближеними даними;

- розробка алгоритмів та програм дослідження достовірності розв’язків, отриманих прямими та ітераційними методами;
- апробація алгоритмів для математичного моделювання в прикладних задачах.

Об’єкт дослідження – математичні моделі, що описуються СЛАР з розрідженими матрицями нерегулярної структури.

Предмет дослідження – паралельні методи та комп’ютерні алгоритми знаходження розв’язку СЛАР з розрідженими матрицями нерегулярної структури.

Методи дослідження. У роботі застосовуються методи теорії матриць, лінійної алгебри, теорії графів, функціонального аналізу, теорії похибок, теорія нейронних мереж.

СЛАР, ПАРАЛЕЛЬНІ ОБЧИСЛЕННЯ, РОСПІЗНАВАННЯ ЗОБРАЖЕНЬ, НЕЙРОННА МЕРЕЖА, КЛАСИФІКАЦІЇ ДАНИХ, НЕНУЛЬОВІ ЕЛЕМЕНТИ

ABSTRACT

Now one of the main stages in the study of objects, phenomena and processes of different nature is mathematical modeling and related computer experiment. Numerous experiments give an opportunity to plan a full-scale experiment, as well as to get new knowledge about those processes and phenomena for which it is difficult, or in general, impossible to carry out a full-scale experiment. A large number of mathematical models can be described by systems of linear algebraic equations (SLRs) with soldered matrices after performing the corresponding transformations.

The main feature of such systems is their large orders and a small number of non-zero elements. Large orders of SLAR arise due to the fact that researchers want to get the most reliable results, which is why more detailed models are being built. The small number of non-zero elements is due to the discretization of the model. In particular, systems of equations with sparse matrices arise in problems of analysis of the strength of structures in civil and industrial construction, filtration, heat and mass transfer, and others like that. Scope of the methods of solving SLR with sparse matrices is constantly expanding. Because of this, there is an interest in the problem of constructing effective methods for solving such systems, whose orders exceed hundreds of thousands.

Classical results concerning the development of methods for solving SLRR with rarefied matrices are covered in a series of monographs of American and domestic authors: A. George, J. Liu, S. Pisanetski, J. Golub, R. Tjurson, I. A. Blatova, ME Ekseryrovskaya and others.

Also, the requirements for the computer technology used to conduct a computer experiment are growing. It must provide sufficient speed and have the required amount of resources so that the result of the experiment can be obtained over a relatively short period of time. Now in the market there are many different architectures of computers with parallel computing organization. The most productive are the platforms of the so-called "hybrid" architecture. These systems combine MIMD (multiple instructions - multiple data) and SIMD architecture (single instruction - multiple data), in particular, in a multi-core processor system,

computations are accelerated by means of a graphical accelerator. Hence, one of the effective approaches to solving SLR with sparse matrices is the construction of parallel algorithms that take into account the peculiarities of computer architecture.

The main problems of developing effective parallel algorithms are: analysis of the structure of the matrix, or bringing it to the corresponding form, using appropriate conversion algorithms; choice of effective data decomposition; determining the effective number of processor cores and graphic accelerators used for calculations; definition of the interprocess communication topology, which reduces the number of communications and synchronizations.

It is precisely for analyzing the structure of a sparse matrix that a neural network is used which allows the selection of groups of non-zero elements that can be processed independently. The results of the analysis will be based on the decomposition of data and the number of computing cores to be selected, which will provide the shortest settlement time for a particular matrix structure.

The purpose and objectives of the study. The purpose of the work is to develop and research parallel methods and computer algorithms for research and solving SLR with sparse matrices of irregular structure on computers of MIMD architecture and MIMD and SIMD architecture combinations, testing of algorithms in mathematical modeling in applied problems.

The research tasks include:

• development and research of iterative parallel algorithms for SLR with sparse matrices of irregular structure with approximate data;

• development of algorithms and programs for investigating the validity of solutions obtained by direct and iterative methods;

• Approbation of algorithms for mathematical modeling in applied problems.

The object of the study is the mathematical models described by SLAR with sparse matrices of the irregular structure.

The subject of the study is parallel methods and computer algorithms for locating the SLR solution with sparse matrices of the irregular structure.

Research methods. The paper uses methods of matrix theory, linear algebra, graph theory, functional analysis, error theory, and the theory of neural networks.

SLAR, PARALLEL CALCULATION, IMAGE RISK, NEURAL NETWORK, DATA CLASSIFICATION, NONZERO ELEMENTS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОСИЛАНЬ ТА ТЕРМІНІВ

Умовні скорочення

МСЕ – метод скінчених елементів.

СЛАР – система лінійних алгебраїчних рівнянь.

Терміни

Розмірність матриці – кількість невідомих в матриці.

Розріджена матриця – матриця, в якій більшість елементів дорівнює нулю.

Оперативна пам'ять – енергозалежний вид пам'яті комп'ютера, в якій зберігаються дані, необхідні процесору в даний час.

ВСТУП

Актуальність дослідження

Зараз одним з основних етапів при дослідженні об’єктів, явищ і процесів різної природи є математичне моделювання і пов’язаний ним комп’ютерний експеримент. Чисельні експерименти дають можливість, як планувати натурний експеримент, так і отримувати нові знання про ті процеси і явища для яких утруднений, або взагалі неможливий натурний експеримент. Велика кількість математичних моделей після виконання відповідних перетворень можуть бути описанні системами лінійних алгебраїчних рівнянь (СЛАР) з розрідженими матрицями.

Основною особливістю таких систем є їхні великі порядки і невелика кількість ненульових елементів. Великі порядки СЛАР виникають за рахунок того, що дослідники хочуть отримати якомога достовірніші результати, через це будуються більш деталізовані моделі. Мала кількість ненульових елементів пояснюється особливостями дискретизації моделі. Зокрема, системи рівнянь з розрідженими матрицями виникають у задачах аналізу міцності конструкцій у цивільному та промисловому будівництві, фільтрації, тепло- та масо переносу, тощо. Область застосування методів розв’язування СЛАР з розрідженими матрицями постійно розширюється. Через це виникає інтерес до проблеми побудови ефективних методів розв’язання таких систем, порядки яких перевищую сотні тисяч.

Класичні результати, що стосуються розробки методів розв’язання СЛАР з розрідженими матрицями висвітлюються у ряді монографій американських і вітчизняних авторів: А. Джорджа [1], Дж. Лю [2], С. Писанецьки [18], Дж. Голуба [3], Р. Тюарсона [4], І.А. Блатова [5], М.Е. Ексаревської та інших. Також зростають вимоги до обчислювальної техніки, що використовується для проведення комп’ютерного експерименту. Вона повинна забезпечувати достатню швидкодію і мати необхідну кількість ресурсів, щоб результат експерименту можна було отримати за досить невеликий проміжок часу. Зараз

на ринку представлені багато різних архітектур комп’ютерів з паралельною організацією обчислень. Найбільш продуктивними є платформи так званої «гібридної» архітектури. Дані системи поєднують у собі MIMD- (multiple instructions – multiple data) та SIMD-архітектури (single instruction – multiple data), а саме у системі з багатоядерними процесорами обчислення прискорюються за рахунок графічного прискорювача. Отже одним з ефективних підходів до розв’язання СЛАР з розрідженими матрицями є побудова паралельних алгоритмів, що враховують особливості архітектури комп’ютера.

Основними проблемами розробки ефективних паралельних алгоритмів є: аналіз структури матриці, або приведення її до відповідного вигляду, застосовуючи відповідні алгоритми перетворення; вибір ефективної декомпозиції даних; визначення ефективної кількості процесорних ядер і графічних прискорювачів, що використовуються для обчислень; визначення топології міжпроцесних зв’язків, яка зменшує кількість комунікацій і синхронізацій.

Саме для аналізу структури розрідженої матриці використовується нейрона мережа, яка дозволить виділити групи ненульових елементів, які можуть оброблятись незалежно. За результатами аналізу буде будуватись декомпозиція даних та обиратись кількість обчислювальних ядер, що забезпечить найкоротший час розрахунків для конкретної структури матриці. Це забезпечить автоматизацію підготовки даних (а саме матриці) до безпосереднього обчислення та розрахунку СЛАР, що у свою чергу зменшить загальній час необхідний для обрахування обраної проблеми.

Мета та завдання дослідження

Метою роботи є розробка та дослідження паралельних методів та комп’ютерних алгоритмів для дослідження та розв’язування СЛАР з розрідженими матрицями нерегулярної структури на комп’ютерах MIMD-архітектури та комбінації MIMD- і SIMD-архітектури, апробація алгоритмів

при математичному моделюванні у прикладних задачах, розробка та дослідження нейронної мережі для автоматизації декомпозиції даних.
До завдань дослідження належать:

- розробка та дослідження ітераційних паралельних алгоритмів для СЛАР з розрідженими матрицями нерегулярної структури з наближеними даними;
- розробка алгоритмів та програм дослідження достовірності розв’язків, отриманих прямими та ітераційними методами;
- апробація алгоритмів для математичного моделювання в прикладних задачах;
- розробка та дослідження нейронної мережі для аналізу ненульових елементів розріджених матриць СЛАР.

Об’єкт дослідження

Об’єкт дослідження – математичні моделі, що описуються СЛАР з розрідженими матрицями нерегулярної структури.

Предмет дослідження

Предмет дослідження – паралельні методи та комп’ютерні алгоритми знаходження розв’язку СЛАР з розрідженими матрицями нерегулярної структури.

Методи дослідження

Методи дослідження. У роботі застосовуються методи теорії матриць, лінійної алгебри, теорії графів, функціонального аналізу, теорії похибок, теорія нейронних мереж.

1 АНАЛІЗ ПРОБЛЕМ ВИРІШЕННЯ СЛАР В ЗАДАЧАХ МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ

1.1 Вирішення СЛАР в задачах математичного моделювання

На сьогоднішній день є багато задач та проблем які потребують комп’ютерного моделювання. Це у свою чергу потребує побудови математичних моделей. При більш точному описанні модельованого об'єкта або явища ускладнюся також математичні моделі які базуються на них.

Комп'ютерне моделювання, що виникло як один із напрямів математичного моделювання, з розвитком комп'ютерних технологій стало самостійною і важливою областю застосування комп'ютерів [6]. Комп'ютерні моделі використовуються для вирішення завдань про об'єкти, що моделюються. При роботі з моделлю завдання про моделюється об'єкті може бути сформульована у вигляді мети, т. е. як завдання набуття та вивчення бажаного стану моделі. Постановка мети передбачає визначення складу та сутності конкретного об'єкта, його структури, основних властивостей і взаємодії з навколишнім світом - розуміння моделі, а також цілеспрямоване втручання в функціонування моделі - управління моделлю [7].

У наш час комп’ютерне моделювання застосовується у багатьох областях. Комп'ютерні моделі стали звичайним інструментом математичного моделювання і застосовуються у фізиці, астрофізиці, механіці, хімії, біології, економіці, соціології, метеорології, інших науках і прикладних задачах в різних областях радіоелектроніки, машинобудування, автомобілебудування та ін [8]. Комп'ютерні моделі використовуються для отримання нових знань про моделюється об'єкті або для наближеної оцінки поведінки систем, занадто складних для аналітичного дослідження. Різні сфери застосування комп'ютерних моделей висувають різні вимоги до надійності одержуваних з їх допомогою результатів. Для моделювання будівель і деталей літаків потрібна висока точність і ступінь достовірності, тоді як моделі еволюції міст і

соціально-економічних систем використовуються моделі для отримання наближених або якісних результатів для аналітичного дослідження.

Велика кількість наведених вище математичних моделей, після необхідних перетворень, можуть бути описанні у вигляді деякої СЛАР. При чому найчастіше матриці СЛАР представленні у розрідженому вигляді. При вирішенні точних задач з багатьма змінними СЛАР набувають такої складності при якій доцільно використовувати ЕОМ та спеціальне програмне забезпечення для їх вирішення.

Тому розробка ефективних підходів знаходження розв'язків систем рівнянь є надзвичайно актуальною проблемою, вирішенню якої приділяють чимало уваги.

1.2 Огляд літератури

Значна кількість досліджень проводиться для систем з розрідженими матрицями, з якими пов'язано немало практичних застосувань. На даний момент найбільш повно питання, пов'язані з розв'язуванням СЛАР з розрідженими матрицями, висвітлено у вітчизняних [9, 10, 11, 12, 13, 14, 15] та зарубіжних працях [16, 3,17, 18, 4, 19].

Монографія Тьюарсона [4] є однією з перших у світовій літературі та висвітлює ряд фундаментальних положень роботи з розрідженими матрицями. У монографії С. Писанецки [18] розглянуто як методи розв'язування СЛАР, так і формати зберігання даних та алгоритми попередньої оптимізації впорядкування. У роботах [14, 3] проведено ґрунтовні дослідження питань пов'язаних з форматами та попередньою обробкою розріджених даних. Монографія [20] вважається однією з фундаментальних щодо методів розв'язування СЛАР з додатно визначеними розрідженими матрицями великих порядків. Монографія [19] присвячена розробці прямих методів. У роботі [21] подано надзвичайно широкий опис з теоретичним обґрунтуванням різних класів ітераційних методів розв'язування СЛАР. Монографія [9] описує ітераційні методи з передобумовленням з допомогою неповної факторизації.

Роботи [1, 22] дають змогу провести порівняльний аналіз методів для щільних та розріджених матриць. У [23] запропоновано ефективні обчислювальні схеми для СЛАР з λ-матрицями. Отже, на даний момент існує потужна методична база загальних методів. Проте, як правило, на практиці нерідко задачі мають особливості, використання інформації про які дозволяє будувати ще більш ефективні схеми.

Ключовим питанням, яке виникає під час розробки методів для розріджених систем, є вибір формату зберігання даних. У залежності від специфіки задачі та особливості організації даних можна скористатися вже розробленими схемами [16, 9, 12, 24, 19].

Серед важливих задач побудови ефективних методів для задач з розрідженими матрицями слід окремо виділити задачу оптимізації заповнення. У світі розроблено низку методів, які дозволяють змінити профіль розрідженої матриці за вибраним критерієм [16, 12, 17, 18].

При вирішенні задачі розв’язування СЛАР з розрідженими матрицями можна скористатися одним з двох підходів. Перший полягає у використанні прямих методів. Цей напрямок представлено досить широко як у новітніх, так і у фундаментальних розробках, зокрема, у працях [25, 17, 26, 27, 28, 29, 30, 19]. Другий підхід пропонують ітераційні методи, розробці яких присвячено ряд робіт [31, 32, 16, 9, 33, 34, 35, 36, 24].

Складність розв’язуваних на комп’ютері задач та об’єм інформації, яку необхідно обробляти при розв’язуванні різних класів науково- технічних задач, у тому числі й у випадку СЛАР, постійно зростають. Попри швидке збільшення продуктивності комп’ютерів за рахунок розвитку елементної бази, комп’ютери з одним потоком команд і даних (типу SISD за класифікацією Фліна [37]) не завжди здатні забезпечити необхідні обчислювальні ресурси.

Одним з резервів підвищення продуктивності обчислювальної техніки є використання паралельних комп’ютерів гібридної архітектури. На сьогодні такі комп’ютери виступають основним засобом для природничих наукових досліджень. Аналіз швидкодії комп’ютерів, які використовуються в

різноманітних областях діяльності, показує, що перші місця в списку високопродуктивних комп’ютерів top500 [38] займають саме комп'ютери гібридної архітектури. Вони широко використовуються при моделюванні складних процесів та явищ у геології, хімії, фармакології, біології, медицині, метеорології та інших галузях сучасних природничих наук. Досить широке застосування комп’ютери гібридної архітектури знаходять у машинобудуванні, будівництві, екології, ядерній енергетиці. Лише врахування архітектури комп’ютерів, їх функціональних можливостей та технічних характеристик комунікаційної мережі дають змогу отримати високу експлуатаційну продуктивність мультипроцесорних комп’ютерів. Тому розробка нових алгоритмів або розпаралелювання існуючих для реалізації на паралельних комп’ютерах певного класу виступають актуальними задачами.

Огляд літератури по паралельним обчисленням у лінійній алгебрі проведено у [39, 40]. Окрім великої кількості статей паралельним процесам присвячено ряд монографій [41, 42, 43, 44], де наводиться обширна бібліографія щодо цього питання. Так, наприклад, бібліографія монографії [45] містить 1030 найменувань.

Монографія [46] присвячена побудові та дослідженню паралельних алгоритмів розв’язування задач лінійної алгебри на паралельних обчислювальних системах визначеної архітектури. У [47] подано систематизоване викладення чисельних алгоритмів та організацію паралельних обчислювальних систем. Робота [48] містить математичні основи розробки асинхронних обчислювальних методів, призначених для реалізації на високопродуктивних паралельних обчислювальних системах. Монографія [49] присвячена побудові та дослідженню паралельних алгоритмів для розв’язування задач обчислювальної математики на комп’ютерах з множинним потоком команд та даних (MIMD- комп’ютерах). У [50] викладено принципи організації паралельних обчислень для розв’язування СЛАР на векторних та паралельних комп’ютерах. У роботах [41, 51] описано операційні системи та мови, які підтримують паралельні обчислення, обговорюються питання

побудови паралельних алгоритмів для задач чисельного аналізу. У [43] викладено паралельні алгоритми розв’язування задач лінійної алгебри. У монографії [29] розроблено та досліджено паралельні алгоритми розв’язування основних класів задач обчислювальної математики на паралельних комп’ютерах. Монографія [30] описує їх практичну реалізацію на інтелектуальному паралельному комп’ютері Інпарком. Автори монографії [42] розглядають програмне середовище та його функціональні можливості для організації паралельних обчислень на мережі комп’ютерів. Інформацію про різні бібліотеки стандартних паралельних програм подано в [52].

Окремо значну групу досліджень становлять розробки алгоритмів для ітераційних методів розв’язування СЛАР. Найбільш фундаментальний огляд питань, пов’язаних з ітераційними методами розв’язування СЛАР з розрідженими матрицями, подано в монографії [21]. У ній наведено розгорнутий огляд базових понять лінійної алгебри, методи дискретизації, теорію розріджених матриць, повний перелік різних класів ітераційних методів, способи передобумовлення та питання паралельної реалізації. У доповнення монографії автори опублікували ряд статей: передобумовлення на основі неповної факторизації [31], різні способи покращення неповної факторизації [32], побудова відповідного програмного забезпечення [53] та інші (всього близько 15 базових питань, наведених у монографії).

Монографія [9] присвячена розробці ітераційних методів з передобумовлювачем на основі неповної факторизації з теоретичним обґрунтуванням вибору методів. Також питання, присвячені неповній факторизації, висвітлюються в роботі [33], яка підкріплена чисельними експериментальними даними. Монографія [36] присвячена розробці ітераційних методів для розв’язування задач мікроелектроніки. Робота [54] пропонує реалізацію методу ітерацій Чебишева з використанням графічних процесорів.

У працях [23, 55] проводиться порівняння прямих та ітераційних методів розв’язування СЛАР з розрідженими матрицями на прикладі задач, що

виникають при моделюванні будівельних конструкцій. Методи розв’язування СЛАР зі стрічковою матрицею розглядаються в роботі [13].

Дослідження [35] присвячене питанням побудови передобумовлювача на багатопроцесорних системах. У ньому розглядається питання ефективної побудови методу на основі розвинення Холецького. У роботах [56, 34] описується паралельна реалізація ітераційного методу з передобумовлювачем на основі неповної факторизації. Блочно-діагональний варіант цього методу розглядається у праці [57]. У статті [24] описується використання паралельного ітераційного методу з передобумовлювачем на основі методів вкладених перерізів у задачах гідродинаміки. У роботах [58, 54, 22] розглядаються питання реалізації на графічних прискорювачах ітераційних методів з передобумовлювачами.

Готові рішення щодо розв’язування СЛАР з розрідженими матрицями пропонують програмні пакети [59, 60], де програми побудовані на основі ітераційних методів. Програмний комплекс [61] пропонує широкий вибір методів розв’язування систем із симетричними матрицями. Бібліотека програм [59] побудована на основі методів, описаних у монографії [16].

Проте, не зважаючи на значну кількість досліджень, присвячених побудові паралельних алгоритмів швидкозбіжних ітераційних методів розв’язування СЛАР, проблема лишається актуальною. Досі не розроблено методу, який би ефективно розв’язував задачу з матрицею будь-якої структури. Крім того, у ряді випадків ефективність розпаралелювання вдається досягти лише на невеликій кількості процесів [34].

Як показує досвід, необхідною умовою створення ефективних паралельних алгоритмів і програм для розв’язування СЛАР з розрідженими матрицями є врахування архітектури паралельних комп’ютерів та особливостей структури розрідженої матриці з метою забезпечення високої ефективності алгоритмів та збалансованості обчислювальних процесів.

1.3 Огляд наявних реалізацій

Проблеми розвитку прикладного програмного забезпечення для розв’язування СЛАР та супутніх задач лінійної алгебри знаходяться у центрі уваги багатьох спеціалістів із часів розробки перших комп’ютерів. Тут можна зазначити роботи [62, 63, 64, 49, 65, 66]. Появу в кінці 60-х алгол-процедур, розроблених Вілкінсоном та Райншем [67], можна віднести до початку створення математичного забезпечення лінійної алгебри. Алгол-процедури ввійшли до складу бібліотек прикладних програм WR [68]. Ці процедури в подальшому лягли в основу багатьох бібліотек та пакетів прикладних програм з лінійної алгебри.

Серед найвідоміших бібліотек, які витримали випробовування часом – бібліотека SSP фірми IBM [69], Boeing Library [60], Bell Laboratories [70], бібліотека IMSL (International Mathematical and Statistical Libraries) [67], бібліотека NAG (Numerical Algorithms Group) [71]. Великою кількістю підпрограм у них реалізовано клас задач з лінійної алгебри.

Також варто зазначити, що останнім часом завдяки постійному вдосконаленню обчислювальних характеристик багатопроцесорних комплексів набувають актуальності питання, пов’язані з розробкою паралельних методів розв’язування систем рівнянь [68]. У цьому напрямку розроблено чимало нових паралельних програмних засобів для розв’язування власне СЛАР [72, 73, 31, 74, 75, 76, 77, 47, 32, 16, 21, 53, 78]. Більшість з них орієнтовані на розв’язування певних класів задач, що впливає на набір методів, які вони пропонують. Наприклад, у пакетах BlockSolver95 [73] і SuperLU [21] реалізовано прямі методи розв’язування СЛАР. Ряд інших програмних комплексів, наприклад, pARMS [32], пропонують ітераційний підхід до вирішення проблеми. Оскільки більшість задач мають прикладний характер, значна частина програм орієнтована на СЛАР певного вигляду. Так, у задачах у будівельній, машинобудівельній галузі матриці коефіцієнтів систем зазвичай мають розріджену структуру. Саме для матриць розрідженої структури розроблено низку як прямих (наприклад, у пакетах MUMPS [61], PSPASES [77]), так і

ітераційних (наприклад, пакети PSparslib [59], SPARSKIT [79], Paralution [78]) методів.

Методи розв’язування СЛАР знаходять своє застосування в електроніці, будівництві, машинобудуванні та багатьох інших прикладних областях [25, 23, 24, 80, 37, 81]. А, оскільки кожна окрема галузь застосування висуває свої вимоги до задач, то, не дивлячись на досить широкий вибір готових рішень, виникає потреба в розробці спеціалізованого програмного забезпечення, орієнтованого на певний клас задач.

Отже основною проблемою вирішення СЛАР з розрідженими матрицями є врахування особливостей їх структури та архітектури паралельних комп’ютерів задля забезпечення збалансованості обчислювальних процесів та ефективності алгоритмів.

Тому метою роботи є розробка та дослідження паралельних методів та комп’ютерних алгоритмів для дослідження та розв’язування СЛАР з розрідженими матрицями нерегулярної структури на комп’ютерах MIMD-архітектури та комбінації MIMD- і SIMD-архітектури, апробація алгоритмів при математичному моделюванні у прикладних задачах, розробка та дослідження нейронної мережі для автоматизації декомпозиції даних.

Задачами розробки є:

- розробка та дослідження ітераційних паралельних алгоритмів для СЛАР з розрідженими матрицями нерегулярної структури з наближеними даними;
- розробка алгоритмів та програм дослідження достовірності розв’язків, отриманих прямими та ітераційними методами;
- апробація алгоритмів для математичного моделювання в прикладних задачах;
- розробка та дослідження нейронної мережі для аналізу ненульових елементів розріджених матриць СЛАР.

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ РОЗВ'ЯЗАННЯ СЛАР З РОЗРІДЖЕННИМИ МАТРИЦЯМИ

2.1 Огляд проблеми

Перед формулювання проблеми необхідно оглянути загальні методи вирішення СЛАР з розрідженими матрицями та аналізувати способи впровадження покращень на базі нейромережі.

2.2 Точні методи розв'язку СЛАР

Розглянемо ряд точних методів розв'язку СЛАР [4,5].

2.2.1 Метод Гауса

Нехай дана система $A\bar{x} = \bar{b}$, де A – матриця розмірності $m \times m$ (квадратна). У припущенні, що $a_{11} \neq 0$, перше рівняння системи

$$\sum_{j=1}^{m} a_{ij} x_j = a_{i,m+1}, i = 1, \dots, m \tag{1.1}$$

ділимо на коефіцієнт a_{11}, у результаті одержуємо рівняння:

$$x_1 + \sum_{j=2}^{m} \frac{a_{1j}}{a_{11}} x_j = \frac{a_{1,m+1}}{a_{11}}. \tag{1.2}$$

Потім з кожного з інших рівнянь віднімається перше рівняння, помножене на відповідний коефіцієнт a_{i1}. У результаті ці рівняння перетворяться до виду:

$$\sum_{j=2}^{m} a_{ij}^{1} x_j = a_{i,m+1}^{1}, i = 2, \dots, m. \quad (1.3)$$

Перше невідоме виявилося виключеним із усіх рівнянь, крім першого. Далі в припущенні, що $a_{22}^{1} \neq 0$, ділимо друге рівняння на коефіцієнт a_{22}^{1} і виключаємо невідоме з усіх рівнянь, починаючи з третього і т.д. У результаті послідовного виключення невідомих, матриця системи рівнянь стане трикутною:

$$x_i + \sum_{j=i+1}^{m} a_{ij}^{i} x_j = a_{i,m+1}^{i}, i = 1, \dots, m. \quad (1.4)$$

Сукупність проведених обчислень називається прямим ходом методу Гауса.

З *m*-го рівняння системи визначаємо x_m, з *(m-1)*-го рівняння визначаємо x_{m-1} і т.д. до x_1. Сукупність таких обчислень називають зворотнім ходом методу Гауса.

Реалізація прямого методу Гауса вимагає $N \sim 2m^2/3$ арифметичних операцій, а зворотнього – $N \sim m^2$ арифметичних операцій.

2.2.2 Метод Крамера

Нехай дана система лінійних рівнянь, в якій число рівнянь дорівнює числу невідомих:

(1.5)

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, i = 1, \dots, n.$$

Припустимо, що визначник системи $d \neq 0$. Якщо тепер замінити послідовно у визначнику стовпці коефіцієнтів при невідомих x_j стовпцем вільних членів b_i, то вийдуть відповідно n визначників $d_1,\dots,d_n$.

Теорема Крамера. Система n лінійних рівнянь з n невідомими, визначник якої відмінний від нуля, завжди сумісна і має єдиний розв'язок, який обчислюється по формулах:

$$x_1 = \frac{d_1}{d}; x_2 = \frac{d_2}{d}; \dots; x_{n-1} = \frac{d_{n-1}}{d}; x_n = \frac{d_n}{d}. \quad (1.6)$$

Розв'язок довільних систем лінійних рівнянь. Нехай

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, i = 1, \dots, m \quad (1.7)$$

довільна система лінійних рівнянь, де число m рівнянь системи менше числа n невідомих. Тоді в матриці СЛАР знайдуться $r \leq m$ лінійно незалежних рядків, а інші m-r рядків виявляться їхніми лінійними комбінаціями. Перестановкою рівнянь можна добитися того, що ці r лінійно незалежних рядків займуть перші r місць.

Звідси випливає, що кожне з останніх m-r рівнянь системи (1.7) можна представити як суму перших r рівнянь (які називаються лінійно незалежними або базисними), узятих з деякими коефіцієнтами. Тоді система еквівалентна наступній системі r рівнянь з n невідомими:

(1.8)

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, i = 1, \dots, r.$$

Припустимо, що мінор r-ї розмірності, складений з коефіцієнтів при перших r невідомих, відмінний від нуля: $M_r \neq 0$, тобто є базисним мінором. У цьому випадку невідомі, коефіцієнти при яких складають базисний мінор, називаються базисними невідомими, а інші n-r – вільними невідомими.

У кожному з рівнянь системи (1.8) перенесемо в праву частину всі члени з вільними невідомими $x_{r+1},\dots,x_n$. Тоді одержимо систему, яка містить r рівнянь з r базисними невідомими. Оскільки визначник цієї системи є базисний мінор M_r, то система має єдиний розв'язок щодо базисних невідомих, який можна знайти по формулах Крамера. Даючи вільним невідомим довільні числові значення, одержимо загальний розв'язок вихідної системи.

Однорідна система лінійних рівнянь. Нехай дана однорідна система лінійних рівнянь з n невідомими:

$$\sum_{j=1}^{n} a_{ij}x_j = 0, i = 1, \dots, m. \qquad (1.9)$$

Оскільки додавання стовпця з нулів не змінює рангу матриці системи, то на підставі теореми Кронекера-Капеллі ця система завжди сумісна і має, принаймні, нульовий розв'язок. Якщо визначник системи відмінний від нуля і число рівнянь системи дорівнює числу невідомих, то по теоремі Крамера нульовий розв'язок є єдиним.

У тому випадку, коли ранг матриці системи менше числа невідомих, дана система крім нульового розв'язку буде мати і ненульові розв'язки. Для знаходження цих розв'язків у системі (1.9) виділяємо $r<n$ лінійно незалежних рівнянь, інші відкидаємо. У виділених рівняннях у лівій частині залишаємо r базисних невідомих, а інші n-r вільних невідомих переносимо в праву частину.

Тоді приходимо до системи, розв'язуючи яку по формулах Крамера, виразимо r базисних невідомих $x_1,\ldots,x_r$ через n-r вільних невідомих.

2.2.3 Метод головних елементів

Нехай дана система n лінійних рівнянь з n невідомими:

$$\sum_{j=1}^{n} a_{ij}x_j = a_{i,n+1}, i = 1, \ldots, n, \qquad (1.10)$$

де елементи a_{ij} $(i,j=1,\ldots,n)$ утворюють розширену матрицю системи $\bar{A}$.

Виберемо найбільший по модулю і не належачий стовпцю вільних членів елемент a_{pq} матриці $\bar{A}$, який називається головним елементом, і обчислимо множники $m_i=-a_{iq}/a_{pq}$ для всіх рядків з номерами $i\neq p$ (p-й рядок, що містить головний елемент, називається головним рядком).

Далі до кожного другорядного i-го рядку додамо головний рядок, помножений на відповідний множник m_i.

У результаті одержимо нову матрицю, усі елементи q-го стовпця якої, крім a_{pq}, складаються з нулів.

Відкинувши цей стовпець і головний p-й рядок, одержимо нову матрицю, число рядків і стовпців якої на одиницю менше. Повторюємо ті ж операції з отриманою матрицею, після чого одержуємо нову матрицю і т.д.

Таким чином, побудуємо послідовність матриць. Для визначення невідомих x_j поєднуємо в систему всі головні рядки, починаючи з останнього.

Викладений метод розв'язку систем лінійних рівнянь називається методом головних елементів. Необхідна умова його застосування полягає в тому, що визначник матриці не дорівнює нулю [6,7].

2.2.4 Схема Халецького

Нехай система лінійних алгебраїчних рівнянь дана в матричному вигляді: $A\bar{x} = \bar{b}$, де A – квадратна матриця розмірності n; $\bar{x}$, $\bar{b}$ – вектори-стовпці.

Представимо матрицю A у вигляді добутку нижньої трикутної матриці C і верхньої трикутної матриці B з одиничною діагоналлю, тобто $A=CB$, де

$$C = \begin{pmatrix} c_{11} & 0 & 0 & \dots & 0 \\ c_{21} & c_{22} & 0 & \dots & 0 \\ c_{31} & c_{32} & c_{33} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ c_{n1} & c_{n2} & c_{n3} & \dots & c_{nn} \end{pmatrix}, B = \begin{pmatrix} 1 & b_{12} & b_{13} & \dots & b_{1n} \\ 0 & 1 & b_{23} & \dots & b_{2n} \\ 0 & 0 & 1 & \dots & b_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix}.$$

Причому елементи c_{ij} і b_{ij} визначаються по формулах:

$$c_{i1} = a_{i1}, c_{ij} = a_{ij} - \sum_{k=1}^{i-1} c_{ik} b_{kj}, 1 < j \le i; \tag{1.11}$$

$$b_{i1} = \frac{a_{1j}}{c_{11}}, b_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} c_{ik} b_{kj}}{c_{ii}}, 1 < i < j. \tag{1.12}$$

Рівняння $A\bar{x} = \bar{b}$ можна записати в наступному вигляді:

$$CB\bar{x} = \bar{b}. \tag{1.13}$$

Добуток матриці B на вектор-стовпець $\bar{x}$ є вектором-стовпцем, який позначимо через $\bar{y}$:

$$B\bar{x} = \bar{y}. \tag{1.14}$$

Тоді рівняння (1.13) перепишемо у вигляді:

$$C\bar{y} = \bar{b}. \tag{1.15}$$

Тут елементи c_{ij} відомі, тому що матриця A системи $A\bar{x} = \bar{b}$ вважається вже розкладеною на добуток двох трикутних матриць C і B.

Перемноживши матриці в лівій частині рівності (1.15), одержуємо систему рівнянь, з якої одержимо наступні формули для визначення невідомих:

$$y_1 = \frac{a_{1,n+1}}{c_{11}}; y_i = \frac{a_{i,n+1} - \sum_{k=1}^{i-1} c_{ik} y_k}{c_{ij}}, i > 1. \tag{1.16}$$

Невідомі y_i зручно обчислювати разом з елементами b_{ij}.

Після того, як усі y_i визначені по формулах (1.16), підставляємо їх у рівняння (1.14) [8].

Оскільки коефіцієнти b_{ij} визначені в (1.12), то значення невідомих, починаючи з останнього, обчислюємо по наступних формулах:

$$x_n = y_n, x_i = y_i - \sum_{k=i+1}^{n} b_{ik} x_k, i < n. \tag{1.17}$$

2.2.5 Метод квадратного кореня

Даний метод використовується для розв’язку систем лінійних алгебраїчних рівнянь виду

$$A\bar{x} = \bar{b}, \tag{1.18}$$

у яких матриця A симетрична, тобто $A^T=A$, $a_{ij}=a_{ji}$ $(i=j=1,...,n)$.

Розв'язок системи (1.18) здійснюється у два етапи.

Прямий хід. Перетворення матриці A і представлення її у вигляді добутку двох взаємно транспонованих трикутних матриць:

$$A = S^T S, \tag{1.19}$$

де

$$S = \begin{pmatrix} s_{11} & s_{12} & \dots & s_{1n} \\ 0 & s_{22} & \dots & s_{2n} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & s_{nn} \end{pmatrix}, S^T = \begin{pmatrix} s_{11} & 0 & \dots & 0 \\ s_{12} & s_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ s_{1n} & s_{2n} & \dots & s_{nn} \end{pmatrix}.$$

Перемножуючи S^T і S, і дорівнюючи матриці A, одержимо наступні формули для визначення s_{ij}:

(1.20)

$$\begin{cases} s_{11} = \sqrt{a_{11}}, s_{1j} = a_{1j} / s_{11}, j > 1; \\ s_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} s_{ki}^2}, 1 \le i \le n; \\ s_{ij} = \dfrac{a_{ij} - \sum_{k=1}^{i-1} s_{ki} s_{kj}}{s_{ii}}, i < j; \\ s_{ij} = 0, i > j. \end{cases}$$

Після знаходження матриці S систему (1.18) заміняємо двома їй еквівалентними системами з трикутними матрицями (1.19):

$$S^T \bar{y} = \bar{b}, S\bar{x} = \bar{y}. \tag{1.21}$$

Зворотній хід. Записуємо системи (1.21) у розгорнутому виді:

$$\begin{cases} s_{11}y_1 = b_1; \\ s_{12}y_1 + s_{22}y_2 = b_2; \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots; \\ s_{1n}y_1 + s_{2n}y_2 + \cdots + s_{nn}y_n = b_n. \end{cases} \quad (1.22)$$

$$\begin{cases} s_{11}x_1 + s_{12}x_2 + \cdots + s_{1n}x_n = y_1; \\ s_{22}x_2 + \cdots + s_{2n}x_n = y_2; \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots; \\ s_{nn}x_n = y_n. \end{cases} \quad (1.23)$$

Використовуючи (1.22) і (1.23) послідовно знаходимо:

$$y_1 = \frac{b_1}{s_{11}}, y_i = (b_i - \sum_{k=1}^{i-1} s_{ki}y_k)/s_{ii}, i > 1; \quad (1.24)$$

$$x_n = \frac{y_n}{s_{nn}}, x_i = (y_i - \sum_{k=i+1}^{n} s_{ik}x_k)/s_{ii}, i < n. \quad (1.25)$$

2.2.6 Метод прогону

Для розв’язку систем виду $A\bar{x} = F$ або, що те ж саме,

$$A_i x_{i-1} + C_i x_i + B_i x_{i+1} = F_i \quad (1.26)$$

використовується метод прогону, заснований на припущенні, що шукані невідомі зв'язані рекурентним співвідношенням:

$$x_i = \alpha_{i+1}x_{i+1} + \beta_{i+1}, \quad (1.27)$$

де *i=1,…,n-1*.

Використовуючи це співвідношення, виразимо x_{i-1} і x_i через x_{i+1} і підставимо в рівняння (1.26):

$$(A_i\alpha_i\alpha_{i+1} + C_i\alpha_{i+1} + B_i)x_{i+1} + A_i\alpha_i\beta_{i+1} + A_i\beta_i + C_i\beta_{i+1} - F_i = 0, \quad (1.28)$$

де F_i – права частина i-го рівняння. Це співвідношення буде виконуватися незалежно від розв'язку, якщо зажадати

$$A_i\alpha_i\alpha_{i+1} + C_i\alpha_{i+1} + B_i = 0, \quad (1.29)$$

$$A_i\alpha_i\beta_{i+1} + A_i\beta_i + C_i\beta_{i+1} - F_i = 0. \quad (1.30)$$

Звідси випливає:

$$\alpha_{i+1} = \frac{-B_i}{A_i\alpha_i + C_i}, \quad (1.31)$$

$$\beta_{i+1} = \frac{F_i - A_i\beta_i}{A_i\alpha_i + C_i}. \quad (1.32)$$

З першого рівняння одержимо:

$$\alpha_2 = \frac{-B_1}{C_1}, \quad (1.33)$$

$$\beta_2 = \frac{F_1}{C_1}. \quad (1.34)$$

Після знаходження прогоночних коефіцієнтів α і β, використовуючи рівняння (1.27), одержимо розв'язок системи. При цьому

$$x_n = \frac{F_n - A_n\beta_n}{C_n + A_n\alpha_n}. \quad (1.35)$$

2.2.7 Матричний метод

Матричний метод розв'язку систем лінійних алгебраїчних рівнянь з ненульовим визначником полягає в наступному.

Нехай дана система *n* лінійних рівнянь з *n* невідомими (над довільним полем):

$$\begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n = b_1; \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots; \\ a_{n1}x_1 + \cdots + a_{nn}x_n = b_n. \end{cases} \quad (1.36)$$

Тоді її можна переписати в матричній формі: $A\bar{x} = \bar{b}$, де A – основна матриця системи, $\bar{b}$ і $\bar{x}$ – стовпці вільних членів і рішень системи відповідно:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}, \bar{b} = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix}, \bar{x} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}.$$

Помножимо це матричне рівняння ліворуч на A^{-1} – матрицю, зворотню до матриці A:

$$A^{-1}(A\bar{x}) = A^{-1}\bar{b}. \quad (1.37)$$

Оскільки $A^{-1}A=E$ (враховуючи асоціативність матричного добутку), одержуємо $\bar{x} = A^{-1}\bar{b}$. Права частина цього рівняння дасть стовпець рішень

вихідної системи. Умовою застосовності даного методу (як і взагалі існування розв’язку неоднорідної системи лінійних рівнянь із числом рівнянь, рівним числу невідомих) є невиродженість матриці *A*. Необхідною і достатньою умовою цього є нерівність нулю визначника матриці *A*: det *A*≠0.

Для однорідної системи лінійних рівнянь, тобто коли $\bar{b} = 0$, дійсно зворотнє правило: система $A\bar{x} = \bar{0}$ має нетривіальний (тобто ненульовий) розв’язок тільки якщо det *A*=0. Такий зв'язок між рішеннями однорідних і неоднорідних систем лінійних рівнянь зветься альтернативою Фредгольма.

До прямих методів, що використовують властивість розрідженості матриці, можна віднести: алгоритм мінімального ступеня, алгоритм мінімального дефіциту, деревовидна блокова розбивка для асиметричного розкладання, методи вкладених або паралельних перетинів і ін.

2.3 Ітераційні методи розв’язку СЛАР

Наближені методи розв’язку систем лінійних рівнянь дозволяють одержувати значення коренів системи із заданою точністю у вигляді границі послідовності деяких векторів. Процес побудови такої послідовності називається ітераційним (повторюваним).

Ефективність застосування наближених методів залежить від вибору початкового вектора і швидкості збіжності процесу.

2.3.1 Метод простих ітерацій

Нехай дана система *n* лінійних рівнянь з *n* невідомими: $A\bar{x} = \bar{b}$. Припускаючи, що діагональні елементи $a_{ii} \neq 0$ $(i=1,...,n)$, виразимо x_1 через перше рівняння системи, x_2 – через друге рівняння і т.д. У результаті одержимо систему:

$$\begin{cases} x_1 = \dfrac{b_1}{a_{11}} - \dfrac{a_{12}}{a_{11}}x_2 - \dfrac{a_{13}}{a_{11}}x_3 - \dots - \dfrac{a_{1n}}{a_{11}}x_n; \\ x_2 = \dfrac{b_2}{a_{22}} - \dfrac{a_{21}}{a_{22}}x_1 - \dfrac{a_{23}}{a_{22}}x_3 - \dots - \dfrac{a_{2n}}{a_{22}}x_n; \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots; \\ x_n = \dfrac{b_n}{a_{nn}} - \dfrac{a_{n1}}{a_{nn}}x_1 - \dfrac{a_{n2}}{a_{nn}}x_2 - \dots - \dfrac{a_{n,n-1}}{a_{nn}}x_{n-1}. \end{cases} \quad (1.38)$$

Позначимо $b_i/a_{ij}=\beta_i-a_{ij}/a_{ii}=\alpha_{ij}$, де $i,j=1,\dots,n$. Тоді система запишеться, таким чином, у матричній формі: $x=\beta+\alpha x$. Розв'яжемо цю систему методом простих ітерацій. За нульове наближення приймемо стовпець вільних членів. Кожне $(k+1)$-е наближення обчислюють по формулі:

$$x^{(k+1)} = \beta + \alpha x^{(k)}, k = 0, \dots, n. \quad (1.39)$$

Якщо послідовність наближень $x^{(0)},\dots,x^{(k)}$ має границю $x = \lim_{k\to\infty} x^{(k)}$, то ця границя є розв'язком системи, оскільки в силу властивості границі $\lim_{x\to\infty} x^{(k+1)} = \beta + \alpha \lim_{k\to\infty} x^{(k)}$, тобто $x=\beta+\alpha x$ [4,6].

2.3.2 Метод Зейделя

Метод Зейделя являє собою модифікацію методу простих ітерацій.

Нехай дана СЛАР, приведена до нормального вигляду: $x=\beta+\alpha x$. Вибираємо довільно початкові наближення невідомих і підставляємо в перше рівняння системи. Отримане перше наближення підставляємо в друге рівняння системи і так далі до останнього рівняння. Аналогічно будуємо другі, треті і т.д. ітерації.

Таким чином, припускаючи, що k-і наближення $x_i^{(k)}$ відомі, методом Зейделя будуємо $(k+1)$-і наближення по наступних формулах:

$$x_1^{(k+1)} = \beta_1 + \sum_{j=1}^{n} a_{ij} x_j^{(k)}, \tag{1.40}$$

$$x_2^{(k+1)} = \beta_2 + \alpha_{21} x_1^{(k+1)} + \sum_{j=2}^{n} \alpha_{2j} x_j^{(k)}, \tag{1.41}$$

$$x_n^{(k+1)} = \beta_n + \alpha_{nn} x_n^{(k)} + \sum_{j=1}^{n-1} \alpha_{nj} x_j^{(k+1)}. \tag{1.42}$$

2.3.3 Метод релаксації

Метод релаксації – наближений метод розв’язку систем лінійних рівнянь. Система лінійних рівнянь

$$\begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n = b_1; \\ a_{21}x_1 + \cdots + a_{2n}x_n = b_2; \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots; \\ a_{n1}x_1 + \cdots + a_{nn}x_n = b_n. \end{cases} \tag{1.43}$$

приводиться до виду:

$$\begin{cases} -x_1 + b_{12}x_2 + \cdots + b_{1n}x_n + c_1 = 0; \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots; \\ b_{n1}x_1 + b_{n2}x_2 + \cdots - x_n + c_n = 0. \end{cases} \tag{1.44}$$

де

$$b_{ij} = -\frac{a_{ij}}{a_{ii}}, i \neq j, c_i = \frac{b_i}{a_{ii}}. \tag{1.45}$$

Знаходяться нев'язки R_j:

$$\begin{cases} R_1^{(0)} = c_1 - x_1^{(0)} + \sum_{j=2}^{n} b_{1j} x_j^{(0)}; \\ R_2^{(0)} = c_2 - x_2^{(0)} + \sum_{j=1, j\neq 2}^{n} b_{2j} x_j^{(0)}; \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots; \\ R_n^{(0)} = c_n - x_n^{(0)} + \sum_{j=1}^{n-1} b_{nj} x_j^{(0)}. \end{cases} \quad (1.46)$$

Вибирається початкове наближення $x^{(0)}=0$. На кожному кроці необхідно перетворити на нуль максимальну нев'язку:

$$R_s^{(k)} = \delta x_s^{(k)} \Rightarrow R_s^{(k+1)} = 0, R_i^{(k+1)} = R_i^{(k)} + b_{is}\delta x_s^{(k)}. \quad (1.47)$$

Умова зупинки: $\left|R_j^{(k)}\right| < \varepsilon, \forall j = \overline{1,n}.$

Відповідь знаходиться по формулі:

$$x_i \approx x_i^{(0)} + \sum_j \delta x_i^{(j)}. \quad (1.48)$$

2.3.4 Багатосітковий метод

Багатосітковий метод – метод розв’язку систем лінійних алгебраїчних рівнянь, заснований на використанні послідовності зменшуваних сіток і операторів переходу від однієї сітки до іншої. Сітки будуються на основі більших значень у матриці системи, що дозволяє використовувати цей метод при рішенні еліптичних рівнянь навіть на нерегулярних сітках.

Припустимо, що необхідно розв'язати систему виду:

(1.49)

$$Au = f,$$

де A – $n \times n$ матриця з елементами a_{ij}. Для зручності зіставимо індекси з вузлами сітки, таким чином u_i являє собою значення u у вузлі i. Множину вузлів сітки позначимо як $\Omega=\{1,2,\ldots,n\}$. Основна ідея багатосіткових методів полягає в тому, що помилка, яка не може бути усунута методами релаксації, повинна бути прибрана за допомогою корекції з розв’язку на грубій сітці.

Використовуючи верхній індекс як номер рівня введемо наступні позначення:

Послідовність сіток $\Omega = \Omega^1 \supset \Omega^2 \supset \cdots \supset \Omega^M$, причому

$$\Omega^k = C^k \cup F^k, C^k \cap F^k = \emptyset, \Omega^{k+1} = C^k.$$

Сіткові оператори $A=A^1,A^2,\ldots,A^M$.

Оператори інтерполяції P^k, k=1,2,…,M-1.

Оператори згладжування S^k, k=1,2,…,M-1.

Усі ці компоненти багатосіткового методу будуються на першому етапі, відомому як етап побудови.

Етап побудови.

Дорівняти k=1.

Розділити Ω^k на непересічні множини C^k і F^k.

Дорівняти $\Omega^{k+1}=C^k$.

Побудувати оператор інтерполяції P^k.

Побудувати $A^{k+1}=(P^k)^T A^k P^k$.

Побудувати якщо необхідно S^k.

Якщо сітка Ω^k досить мала, дорівняти M=k+1 і зупинитися. Інакше k=k+1 і перехід на крок 2.

Як тільки етап побудови завершений, може бути визначений рекурсивний цикл побудови розв’язку.

Алгоритм: MGV(A^k, P^k, S^k, u^k, f^k).

Якщо k=M, розв'язати $A^M u^M=f^M$ використовуючи прямий метод.

Інакше.

Застосувати метод релаксації S^k μ_1 раз до $A^k u^k = f^k$.

Зробити корекцію на грубій сітці.

Обчислити $r^k = f^k - A^k u^k$.

Обчислити $r^{k+1} = (P^k)^T r^k$.

Застосувати $MGV(A^{k+1}, P^{k+1}, S^{k+1}, e^{k+1}, r^{k+1})$.

Обновити розв'язок $u^k = u^k + P^k e^k + 1$.

Застосувати метод релаксації S^k μ_2 раз до $A^k u^k = f^k$.

Вищенаведений алгоритм описує $V(\mu_1, \mu_2)$ – цикл.

Вибір послідовності сіток і оператора інтерполяції є найбільш важливим елементом етапу побудови і багато в чому визначає якість багатосіткового методу. Критерієм якості є дві вимірювані величини: фактор збіжності – що показує, наскільки швидко сходиться метод, тобто яка кількість ітерацій потрібна для досягнення заданої точності; складність оператора – визначає кількість операцій і об'єм пам'яті, необхідної для кожної ітерації методу.

Складність оператора C_{op} розраховується як відношення кількості ненульових елементів у всіх матрицях A_k, k=1,2,…,*n* до кількості ненульових елементів у матриці першого рівня $A^1 = A$.

2.3.5 Метод Ланцоша

Для розв'язку СЛАР високої розмірності, матриця коефіцієнтів якої зберігається в компактному нижчеописаному вигляді, найбільш зручним ітераційним методом є метод Ланцоша [4], схема якого має вигляд:

$$s_1 = r_0 = Ax_0 - b, r_i = r_{i-1} - a_i A s_i, x_i = x_{i-1} - a_i s_i, \quad (1.50)$$

$$s_{i+1} = r_i + b_i s_i, i = 1, \dots, n, \quad (1.51)$$

де $a_i=(r_{i-1},r_{i-1})/(As_i,s_i)$, $b_i=(r_i,r_i)/(r_{i-1},r_{i-1})$.

Перевагою даного методу є його висока швидкість збіжності до точного розв’язку. Крім того, доведено, що він має властивість «квадратичного закінчення», тобто для додатньо визначеної матриці можна гарантовано одержати точний розв’язок при кількості ітерацій $k \leq n$. Розмір необхідної пам'яті на кожній ітерації не змінюється, тому що не вимагає перетворення матриці A. За критерій зупинки даного ітераційного процесу звичайно використовують співвідношення:

$$|(As_i, s_i)| \leq \varepsilon, \tag{1.52}$$

де ε – задана точність. За інший критерій збіжності іноді зручніше використовувати середньоквадратичну різницю між рішеннями, отриманими на сусідніх ітераціях:

$$\sqrt{\frac{1}{n}\sum (x_i^{(k)} - x_i^{(k-1)})^2} \leq \varepsilon. \tag{1.53}$$

Середньоквадратичну різницю необхідно контролювати при виконанні кожних k наперед заданих ітерацій.

Недоліком методу Ланцоша є сильна залежність збіжності від точності обчислення напрямів спуску.

Практика показує, що при рішенні СЛАР із сильно розрідженими матрицями коефіцієнтів метод Ланцоша має досить високу збіжність, причому в якості початкового наближення можна брати будь-які числа (наприклад, нулі або праву частину).

2.4 Висновки по розділу

Як можна побачити існує багато методів вирішення систем лінійних алгебраїчних рівнянь з розрідженими матрицями. Але при паралелізації цих методів виникає загальна задача розподілення та розбиття задачі на декілька частин та процесорних ядер. Зазвичай для цього проводиться аналіз задачі та її структури (а саме кількості та щільності ненульових елементів матриці). Для автоматизації процесу аналізу структури розрідженої матриці та її ненульових елементів і була розроблена нейромережа наступної структури.

3 РОЗРОБКА НЕЙРОМЕРЕЖІ ДЛЯ АНАЛІЗУ СТРУКТУРИ РОЗРІДЖЕНИХ МАТРИЦЬ

3.1 Математична модель задачі

Розроблюваний вирішувач для лінійних систем (далі солвер) приймає на вхід наступні данні:

- симетрична розріджена матриця нерегулярної структури вигляду : $Ax = b$
- Представлення матриці у вигляді структурного графіку ненульових елементів

Цільова функція описується як класифікація та знаходження вирішення СЛАР

Вхідне представлення обробляєтеся нейронною мережею згорткового типу (див. далі) для аналізу структури матриці та виявлення груп ненульових елементів які можуть бути обробленні незалежно.

Для знаходження розв’язку розглянемо неявний двокроковий метод (2.1)

$$B\frac{y_{k-1}-y_k}{\tau_{k+1}}+Ay_k=f,\ k=1,2,\ldots,B\in H \tag{2.1}$$

де τ_{k+1} ітераційний параметр.

Як відомо від оператора B залежать як число ітерацій, які потрібно виконати для отримання заданої точності ε ,так і число арифметичних операцій, виконуваних при реалізації однієї ітерації. Зараз розвиток ітераційних методів йде шляхом конструювання економічних операторів. Зокрема, до таких належать оператори, яким відповідають діагональна, тридіагональна, трикутна матриці, а також їх комбінації.

Був використаний метод паралельних перерізів, як найбільш зручний. Внаслідок використання методу паралельних перерізів, матриця матиме вигляд (2.2):

$$\hat{A} = P^{\mathrm{T}} A P = \begin{pmatrix} D_1 & 0 & 0 & \cdots & 0 & C_1 \\ 0 & D_2 & 0 & \cdots & 0 & C_2 \\ 0 & 0 & D_3 & & 0 & C_3 \\ \vdots & \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & & D_{p-1} & C_{p-1} \\ C_1 & C_2 & C_3 & \cdots & C_{p-1} & D_p \end{pmatrix} \quad (2.2)$$

де P – матриця перестановок, а блоки Di та Ci зберігають розріджену структуру. Таким чином, задача розв'язування вихідної системи зведеться до розв'язування системи (2.3)

$$Ax = b, \quad x = P^{\mathrm{T}} x, \quad b = P^{\mathrm{T}} b. \quad (2.3)$$

Для обчислення отриманних результатыв застосуемо выще розглянутий методо Зейделя.

3.2 Метод Зейделя

Розглянемо метод Зейделя для блочно-діагональної матриці з обрамленням вигляду методу паралельних перерізів. Представимо матрицю A у вигляді

$$A = D + L + U, \quad (2.4)$$

а оператор B у трикутному вигляді

$$B = D + L, \quad (2.5)$$

де D:

$$D = \begin{pmatrix} D_1 & & 0 \\ & D_2 & \\ & \ddots & \\ 0 & & D_p \end{pmatrix}, \quad L = \begin{pmatrix} 0 & & 0 \\ \vdots & 0 & \\ \vdots & \ddots & \\ L_{p,1} & \cdots & L_{p,p-1} \quad 0 \end{pmatrix}, \quad U = \begin{pmatrix} 0 & \cdots & \cdots & U_{1,p} \\ & 0 & & \vdots \\ & & \ddots & U_{p-1,p} \\ 0 & & & 0 \end{pmatrix}. \quad (2.7)$$

Ітераційні формули для діагональних блоків крім останнього мають вигляд:

$$x_i^{(k+1)} = \left(D_i + L_{i,i}\right)^{-1} \left[b_i - U_{i,i} x_i^{(k)} - U_{i,p} x_p^{(k)}\right], \; i = 1,2,\ldots, p-1, \quad (2.8)$$

а для останнього:

$$x_p^{(k+1)} = \left(D_p + L_{p,p}\right)^{-1} \left[b_p - \sum_{i=1}^{p-1} \left(L_{p,i}^{-1} x_i^{(k+1)}\right) - U_{p,p} x_p^{(k)} \right]. \quad (2.9)$$

Виходячи з цих формул, а також вважаючи, що кожен GPU містить праві частини bi та вектори xi а також відповідні підматриці D, L та U, паралельний алгоритм можна записати у такому вигляді.

1. Одночасно у всіх GPU обчислюємо $x_i^{(k+1)}$ для всіх діагональних блоків крім останнього;
2. Одночасно обчислюються $\left(L_{p,i}^{-1} x_i^{(k+1)}\right)$ і результати передаються у GPU, в якому знаходиться останній діагональний блок, де обчислюється
$$x_p^{(k+1)} = \left(D_p + L_{p,p}\right)^{-1}\left[\tilde{b} - U_{p,p} x^{(k)}\right].$$
3. GPU у якому знаходиться останній діагональний блок обчислює значення
$$\tilde{b} = b_p - \sum_{i=1}^{p-1}\left(L_{p,i}^{-1} x_i^{(k+1)}\right).$$

3.3 Побудова згорткової нейромережі

Оскільки основною задачею нейронної мережі буде аналіз зображень, то був обраний найбільш пристосований тип нейромережі, а саме згорткова. В результаті була розроблена нейромережа наступної структури зображеної на рис 3.1

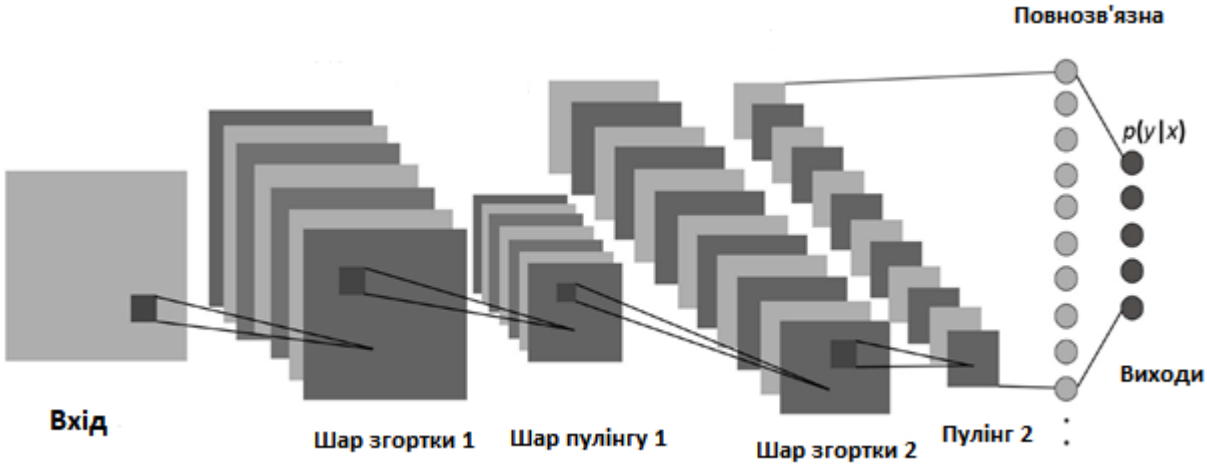

Рисунок 3.1 – Структура згорткової нейромережі.

Згорткова нейронна мережа складається з декількох шарів. Ці шари можуть бути трьох типів:

- Згортковий (Convolutional): Згоркові шари складаються з прямокутної сітки нейронів. Він потребує, щоб попередній шар також був прямокутною сіткою нейронів. Кожен нейрон приймає входи з прямокутного розділу попереднього шару; ваги для цього прямокутного розділу однакові для кожного нейрона в згортковому шарі. Таким чином, згортковий шар - це лише зображена згортка попереднього шару, де вага визначає фільтр згортки. Крім того, у кожному згортковому шарі може бути кілька сіток; кожна сітка приймає вхідні дані всіх сіток попереднього шару, використовуючи потенційно різні фільтри.
- Пулінг (Max-Pooling): після кожного згорткового шару, може існувати шар пулінгу. Пулінговий (агрегувальний) шар приймає невеликі прямокутні блоки з згорткового шару і зменшує його, щоб виготовити єдиний вихід з цього блоку. Існує декілька способів зробити це об'єднання, наприклад, прийняття середнього або максимального, або вивчена лінійна комбінація нейронів у блоці. В данному випадку використовуються шаои макс-пуллінгу. Тобто вони беруть максимум блоку, який вони об'єднують.
- Повноз'єднаний (fully-conected): після декількох згорткових і макс-пулінгових шарів, класифікація високого рівня в нейронній мережі виконуються через повноз'єднанні шари. Повноз'єднаний шар приймає всі нейрони на попередньому шарі (будь то повноз'єднаний, агрегувальний або згортковий) і підключається до кожного окремого нейрона, який він має. Повноз'єднаннї шари більше не розміщені в просторовій структурі тому після повноз'єднанного шару не може бути шарів згортки.

Розроблена нейромережа складається з наступних компонентів або шарів які використовуються декілька разів.

3.3.1 Шар згортки (convolution)

У згорткової нейронної мережі в операції згортки використовується лише обмежена матриця ваг невеликого розміру, яку «рухають» по всьому оброблюваного шару (на самому початку - безпосередньо по вхідному зображенню), формуючи після кожного зсуву сигнал активації для нейрона наступного шару з аналогічною позицією. Тобто для різних нейронів вихідного шару використовуються одна і та ж матриця ваг, яку також називають ядром згортки ... Тоді наступний шар, що вийшов в результаті операції згортки такою матрицею ваг, показує наявність даної ознаки в оброблюваному шарі і її координати, формуючи так звану карту ознак (англ. feature map).

Операція згортки відбувається за наступною формулою:

$$x_{ij}^{l} = \sum_{a=-\infty}^{+\infty} \sum_{b=-\infty}^{+\infty} w_{ab}^{l} \cdot y_{(i \cdot s-a)(j \cdot s-b)}^{l-1} \quad \forall i \in (0, \ldots, N) \quad \forall j \in (0, \ldots, M) \tag{2.10}$$

де підрядкові індекси i, j, a, b — це індекси елементів матриці, а s — величина шагу згортки (stride).

Надрякові l и l−1 — це індекси шарів нейромережі.

xl-1 — вихід попереднього шару, або зображення

yl-1 — це xl-1 після проходження функції активації

wl — ядро згортки

xl — результат операції згортки. Операція згортки проходить окремо по кожному елементу ij матрицы xl, розмірність якої (N,M)(N,M).

При зворотньому методі (backpropagation):

Давайте припустимо, що у нас є деяка функція помилки, E, і ми знаємо значення помилки на нашому згортковому шарі.

Помилка, яку ми знаємо, і що нам потрібно обчислити для попереднього шару, є частковою величиною E по відношенню до кожного виходу нейрона $\left(\frac{\partial E}{\partial y_{ij}^{l}}\right)$. Необхідно з'ясувати, який градієнтний компонент для кожного ваги,

застосовуючи правило ланцюга. В правилі ланцюга ми повинні сумувати внесок усіх виразів, в яких відбувається зміна.

$$\frac{\partial E}{\partial \omega_{ab}} = \sum_{i=0}^{N-m} \sum_{j=0}^{N-m} \frac{\partial E}{\partial x_{ij}^{\ell}} \frac{\partial x_{ij}^{\ell}}{\partial \omega_{ab}} = \sum_{i=0}^{N-m} \sum_{j=0}^{N-m} \frac{\partial E}{\partial x_{ij}^{\ell}} y_{(i+a)(j+b)}^{\ell-1} \tag{2.11}$$

, де l – l тий шар, де l = 1 - перший шар, а l = L - останній шар,

x має розмірність H × W і має i j як ітератори,

фільтр або ядро ω має розмірність $k_1 \times k_2$ має m та n як ітератори

ω_{ab}, - вагова матриця, яка з'єднує нейрони з шаре l з нейронами шару l-1.

x_{ij}^{l} - це зведений вхідний вектор на шарі l, а також зміщення,

представлене як $x_{ij}^{l} = \sum_a \sum_b \omega_{ab}^{l} o_{(i+a)(j+b)}^{l-1} + b^{l}$

У цьому випадку ми повинні сумувати всі вирази x_{ij}^{l}, в яких відбувається ω_{ab}. Ми знаємо, що $\frac{\partial x_{ij}^{l}}{\partial \omega_{ab}} = y_{(i+a)(j+b)}^{l-1}$, просто переглядаючи рівняння прямого поширення (2.10).

Для того, щоб обчислити градієнт, нам потрібно знати значення $\left(\frac{\partial E}{\partial y_{ij}^{l}}\right)$ (які часто називають "дельтами"). Дельти є досить простими для обчислення, знову використовуючи правило ланцюга:

$$\frac{\partial E}{\partial x_{ij}^{\ell}} = \frac{\partial E}{\partial y_{ij}^{\ell}} \frac{\partial y_{ij}^{\ell}}{\partial x_{ij}^{\ell}} = \frac{\partial E}{\partial y_{ij}^{\ell}} \frac{\partial}{\partial x_{ij}^{\ell}} \left(\sigma(x_{ij}^{\ell})\right) = \frac{\partial E}{\partial y_{ij}^{\ell}} \sigma'(x_{ij}^{\ell}) \tag{2.12}$$

Як ми бачимо, оскільки ми вже знаємо помилку на поточному шарі $\frac{\partial E}{\partial y_{ij}^{l}}$, ми можемо дуже легко обчислити дельти $\frac{\partial E}{\partial x_{ij}^{l}}$ на поточному шарі, просто використовуючи похідну від функції активації σ '(x) . Оскільки ми знаємо помилки на поточному шарі, тепер ми маємо все, що потрібно для обчислення градієнта щодо ваг, які використовує цей згортковий шар.

Окрім обчислення ваг для цього згорткового шару, ми повинні поширювати помилки назад до попереднього шару. Ми можемо ще раз використати правило ланцюга:

$$\frac{\partial E}{\partial y_{ij}^{\ell-1}} = \sum_{a=0}^{m-1}\sum_{b=0}^{m-1} \frac{\partial E}{\partial x_{(i-a)(j-b)}^{\ell}} \frac{\partial x_{(i-a)(j-b)}^{\ell}}{\partial y_{ij}^{\ell-1}} = \sum_{a=0}^{m-1}\sum_{b=0}^{m-1} \frac{\partial E}{\partial x_{(i-a)(j-b)}^{\ell}} \omega_{ab} \quad (2.13)$$

Оглядаючись назад у рівняннях прямого поширення, можна сказати, що

$$\frac{\partial x^{l}{}_{(i-a)(j-b)}}{\partial y_{ij}^{l-1}} = \omega ab. \quad (2.14)$$

Це дає нам вище значення для помилки на попередньому шарі. У нас є фільтр ω, який якось застосовується до шару; однак замість x (i + a) (j + b) ми маємо x (i-a) (j-b). Рівняння (2.14) має сенс для точок, які, принаймні, на m віддалені від верхнього та лівого краю. Щоб це виправити, ми повинні поставити верхні та лівими краями нулі.

3.3.2 Шар агрегування (pooling)

Операція агрегування слугує поступовому скороченню просторового розміру представлення для зменшення кількості параметрів та об'єму обчислень у мережі, і відтак також для контролю перенавчання.

Шар пулінгу бере деякий блок k × k і на вихід подає одне значення, яке є максимальним у цьому блоці. Наприклад, якщо на вхід подається шар N × N, вони потім виведуть шар $\frac{N}{k} \times \frac{N}{k}$, так як кожен k × k блок зводиться до лише одного значення за допомогою max функції.

Використовується метод максимального агрегування (maxpooling) із фільтрами розміру 2×2, що застосовуються з кроком 2.

Макс-агрегування має багато переваг:

- Зменшує розмір карти рис та кількість параметрів, що дозволяє запобігти перетренуванню;
- Він зменшує функціональні карти, зберігаючи найважливіші риси;

- Це робить мережу інваріантною для малих перетворень, спотворень та помилок на вхідному зображенні (невелике спотворення на вході не змінить висновок шару пулінгу - оскільки ми беремо максимальне значення найближчих елементів).

3.3.3 Повноз'єднаний шар (fully-conected)

Після шарів згортки ми отримаємо безліч карт ознак. Їх з'єднаємо в один вектор і цей вектор подамо на вхід fully connected мережі.

Формула для fc-слоя (fully connected) виглядає так:

$$x_i^l = \sum_{k=0}^{m} w_{ki}^l y_k^{l-1} + b_i^l \qquad \forall i \in (0, \ldots, n) \tag{2.15}$$

3.3.4 Функція втрат

Завершальний етап мережі - функція, яка оцінює якість роботи всієї моделі (2.12). Функція втрат знаходиться в самому кінці, після всіх шарів мережі.

$$E = \sum_{i=0}^{n} \frac{1}{2}(y_i^{truth} - y_i^l)^2 \tag{2.16}$$

де n — кількість класів, y^l — вивід моделі, а y^{truth} — правильні відповіді.

3.3.5 Відкидання (Dropout)

Відкидання - методика регуляризації для глибоких нейронних мереж. Навіть найсучасніші моделі, які мають точність 95%, збільшують точність на 2%, лише додаючт відкидання, що є досить значним виграшем на цьому рівні.

Функція Dropout використовується для запобігання перетренування (overfitting) при дуже простій ідеї. Під час навчання, при кожній ітерації нейрон тимчасово "скидається" або вимикається з ймовірністю p. Це означає, що всі входи та виходи цього нейрона будуть вимикатися під час поточної ітерації. Відкинуті нейрони відновлюються з ймовірністю p на кожному етапі

тренування, тому відключений нейрон на одному кроці може бути активним у наступному. Гіперпараметр p називається швидкістю відсіву, і це зазвичай становить приблизно 0,5, що відповідає 50% викидних нейронів.

Хоча ми і навмисно вимикаємо нейрони мережа стає працювати краще. Причиною цього є те, що відключення заважає мережі занадто залежати від невеликої кількості нейронів і змушує кожен нейрон працювати самостійно.

Відкидання можна застосувати до вузлів вхідного шару або прихованого шару, але не до вихідних вузлів. Краї з викинутих вузлів вимкнено.Вузли, які були обранні для відключення, змінюються на кожному навчальному етапі. Також відкидання не застосовується під час тестування після тренування мережі, це робиться лише в процесі навчання.

3.3.6 Тренування нейромережі

Для тренування мережі був застосований метод зворотнього поширення помилки (backpropagation).

- Для вузла останнього рівня

$$\delta_j = -o_j(1 - o_j)(t_j - o_j) \quad (2.17)$$

- Для внутрішнього вузла мережі

$$\delta_j = -o_j(1 - o_j) \sum_{k \in Outputs(j)} \delta_k w_{j,k} \quad (2.18)$$

- Для всіх вузлів

$$\Delta w_{i,j} = -\eta \delta_j x_i \quad (2.19)$$

Алгоритм навчання можна розділити на дві фази: розповсюдження та оновлення ваги.

Етап 1: розповсюдження
Кожне поширення передбачає наступні кроки:

1. Розповсюдження вперед через мережу для генерації вихідних значень

2. Розрахунок вартості (термін помилки)
3. Розповсюдження вихідних активацій назад через мережу, використовуючи ціль тренувального шаблону, для генерації дельти (різниця між цільовими та фактичними вихідними значеннями) всіх вихідних і прихованих нейронів.

Етап 2: оновлення ваги
Для кожного ваги слід дотримуватися наступних кроків:

1. Дельта виведення ваги та активація введення множиться, щоб знайти градієнт ваги.
2. Співвідношення (відсоток) градієнта ваги віднімається від ваги.

Це співвідношення (відсоток) впливає на швидкість і якість навчання; це називається швидкістю навчання. Чим більше співвідношення, тим швидше потягує нейрон, але чим менше співвідношення, тим точніше тренування. Знак градієнта ваги вказує на те, чи помилка залежить від ваги безпосередньо або навпаки. Тому вага повинна бути оновлена в зворотному напрямку, "спускаючись" на градієнт.

Навчання повторюється (на нових партіях), поки мережа не виконає належну роботу.

У вигляді псевдокоду алгоритм можна подати наступним чином:

ініціалізувати ваги мережі (часто малі випадкові числа)
do
forEach тренувальні приклади як ex
передбачення = результат-нейромережі(network, ex) // прямий хід
дійсний = результат-прикладу(ex)
обчислити помилку (передбачення - дійсний) на виходах мережі
обчислити {\displaystyle \Delta w_{h}}$\Delta\Delta w_h$ для всіх ваг від внутрішнього до вихідного шару // зворотній хід
обчилити Δw_i для всіх ваг від вхідного до внутрішнього шару // продовження звоторнього ходу

оновити ваги мережі // виключаючи вхідний шар

until усі приклади розпізнано коректно або не досягнутий інший критерій зупинки

return the network

3.4 Висновки до розділу

В цьому розділі був сформульований формальний опис алгоритмів розв’язання задачі та методів побудови і структури використанної нейромережі. В нейромережі були виділенні основні шари та схему використану при їх побудові.

4 ОПИС ПРОГРАМНОГО ТА ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

4.1 Засоби розробки

Розробка проекту проводиться за допомогою мови програмування Python. Дана мова буде зручна як у реалізації алгоритмів так і при створенні частини нейромережі. Інтерфейс та взаємодія користувачів забезпечується через консольний термінал. Реалізація проекту використовуе такі технології, як TensorFlow, CUDA, Keras та інші.

Далі буде розглянуто переваги використаних технологій та мов програмування.

Python. Широко використовувана інтерпретована, динамічна, мова програмування високого рівня. Його філософія дизайну підкреслює читабельність коду, а його синтаксис дозволяє програмістам, висловити поняття в меншій кількості рядків коду, ніж це було б можливо в інших мовах програмування.

Python підтримує кілька парадигм програмування, в тому числі об'єктно-орієнтований, імперативний, функціональний та процедурний підхід до програмування. Він має динамічну систему типів, автоматичне керування пам'яттю, те широку стандартну бібліотеку. Крім цього спільнотою відкритого вихідного коду, розробляються велика кількість допоміжних бібліотек для цієї мови програмування. Однією з таких бібліотек є TensorFlow за допомогою якої реалізована побудова нейромережі.

TensorFlow ™ - це бібліотека програмного забезпечення з відкритим кодом для численних обчислень з високою ефективністю. ЇЇ гнучка архітектура дозволяє легко розгортати обчислення на різних платформах (процесори, графічні процесори, TPU), а також від настільних комп'ютерів до кластерів серверів для мобільних пристроїв. Спочатку розроблена дослідниками та інженерами команди Google Brain в організації AI Google, вона має сильну

підтримку для машинного навчання та глибокого навчання, а гнучкі числові обчислення використовуються в багатьох інших наукових областях.

CUDA - це модель паралельної обчислювальної платформи та інтерфейсу прикладного програмування (API), розроблена компанією Nvidia. Це дозволяє розробникам програмного забезпечення та розробникам програмного забезпечення використовувати графічний процесор (GPU) з підтримкою CUDA для обробки загального призначення - підхід, який називається GPGPU (загальнооб'єдноване обчислення на блоках графічної обробки). Платформа CUDA - це програмний рівень, який забезпечує прямий доступ до набору віртуальних наборів графічних процесорів та паралельних обчислювальних елементів для виконання обчислювальних ядер.

Платформа CUDA призначена для роботи з мовами програмування, такими як C, C ++ і Fortran. Ця доступність спрощує паралельне програмування фахівцями використовувати ресурси GPU, на відміну від попередніх API-інтерфейсів, таких як Direct3D та OpenGL, які вимагають високих навичок графічного програмування. Крім того, CUDA підтримує схеми програмування, такі як OpenACC та OpenCL. Коли він вперше був представлений компанією Nvidia, назва CUDA була абревіатурою Compute Unified Device Architecture, але Nvidia згодом відмовилась від використання акроніму.

Keras - бібліотека нейронних мереж із відкритим кодом, написана на Python. Вона здатна працювати з TensorFlow, Microsoft Cognitive Toolkit, Theano або MXNet. Розроблена для швидкого експерименту з глибокими нейронними мережами, вона зосереджена на тому, щоб бути зручною, модульною та розширюваною. Вона була розроблена як частина дослідницької роботи проекту ONEiros (Open-ended Neuro-Electronic Intelligent Robot Operating System), і її основним автором і супроводжувачем є Франсуа Шолле, інженер Google.

У 2017 році команда Google TensorFlow вирішила підтримати Keras у базовій бібліотеці TensorFlow.Keras задумано як інтерфейс, а не самостійний механізм навчання. Він пропонує більш високий рівень інтуїтивно зрозумілого

набору абстракцій, що полегшує розробку глибоких моделей навчання незалежно від використовуваного обчислювального бекенда.

4.2 Вимоги до технічного забезпечення

4.2.1 Загальні вимоги

Представлений програмний продукт являє собою программу, для роботи якої, необхідна робоча странція з наступним технічним забезпеченням:

а) технічне забезпечення:
   1) процесор з тактовою частотою не нижче 2 ГГц;
   2) об'єм оперативної пам’яті не менше 2 Гб;
   3) ПЗУ типу SSD або HDD об’ємом не менше 20 ГБ;

б) програмне забезпечення:
   1) операційна система – Windows версії не нижче Windows 7;
   2) інтерпретатор Python не нижче версії 3;
   3) Наступні бібліотеки для Python : TensorFlow, scipy, pyplotlib, Keras, pillow.

Для для тренування нейромережі потрібне наступне програмне та технічне забезпечення:

а) комп'ютер з характеристиками не нижче:
   1) процесор з тактовою частотою 2.5 ГГц;
   2) оперативна пам’ять 4 Гб;
   3) ПЗУ типу SSD або HDD об’ємом не менше 20 ГБ;
   4) Відео прискорювач Nvidia з підтримкою технології CUDA (GeForce GTX 590 та краще) .

б) програмне забезпечення:
   1) операційна система – Windows версії не нижче Windows 7;
   2) інтерпретатор Python не нижче версії 3;
   3) Наступні бібліотеки для Python : TensorFlow, scipy, pyplotlib, Keras, pillow;

4) Програмний пакет CUDA.

в) комп’ютерна периферія, до складу якої входить:

1) монітор;

2) мишка;

3) клавіатура.

4.3 Архітектура програмного забезпечення

4.3.1 Опис функціональної моделі

Для проектування діаграми використання спочатку необхідно визначити дійових осіб (акторів), а потім визначити, які дії у системі може виконувати кожен з акторів. В даній системі є два типи дійових осіб – користувачі та система.

Система. Основними функціями системи буде розрахунок та аналіз поданної користувачем інформації.

Користувач. Користувач має змогу вибрати данні для аналізу та обрахунку та визначити типи вихідної інформації.

Тепер визначимо дії, які можуть виконувати актори системи (рисунок 4.1):

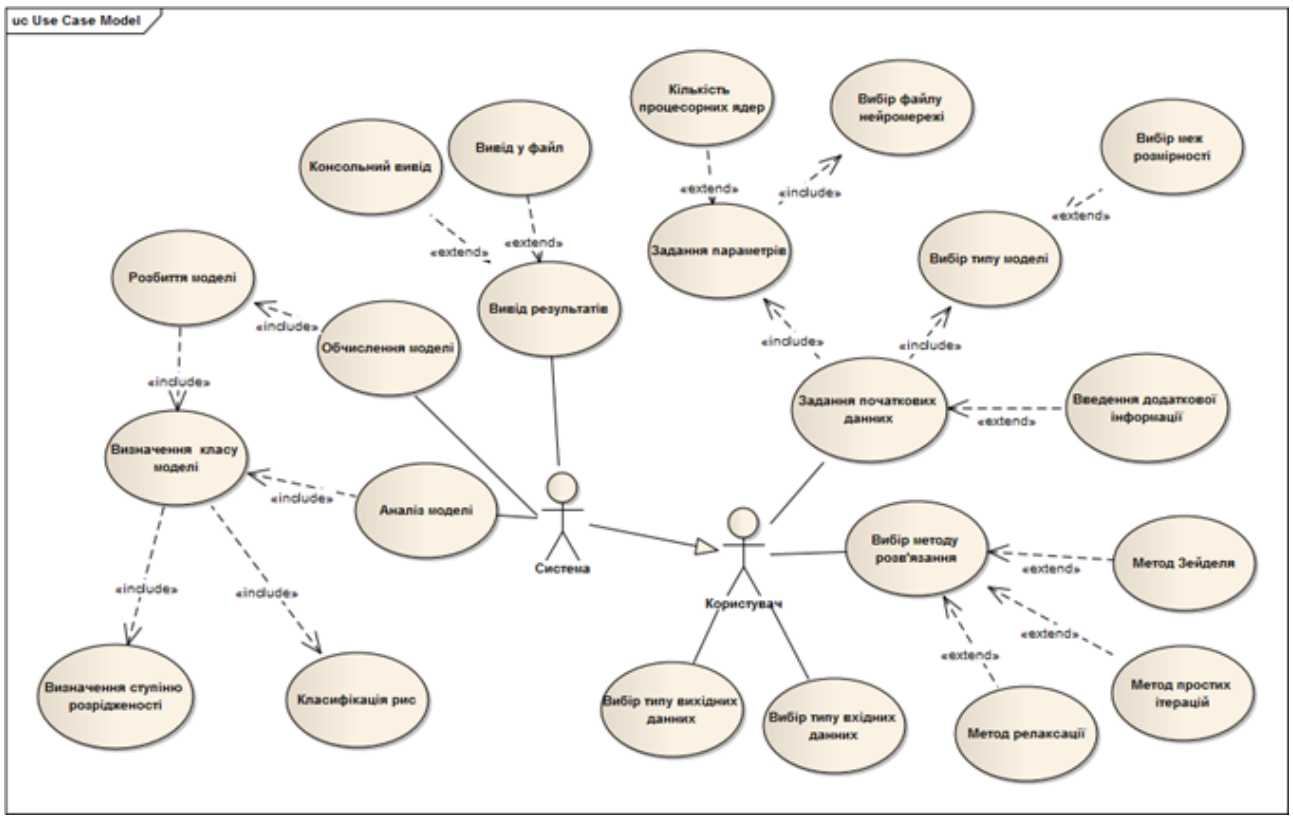

Рисунок 1.3 – Схема структурна варіантів використання. Робота користувача з системою

Список вимог на основі діаграми варіантів використання з їх пріоритетами наведено у таблиці 4.1.

Таблиця 4.1 - Функціональні вимоги

Актор	**Варіант використання**	**Функціональна вимога**	**Пріорітет**
Користувач	Вибір типу вхідних даних	1. Користувач переходить до пункту обрання даних у головному меню. 1.1 Користувач обирає “обрати тип вихідних даних” 1.3. Система відображає доступні типи 1.4 Користувач обирає необхідний тип даних	бажаний

Продовження таблиці 4.1.

Користувач	Вибір типу вхідних даних	2. Користувач переходить до пункту обрання даних у головному меню. 2.1. Користувач обирає “обрати тип вхідних даних” 2.2. Система відображає доступні типи 2.3. Користувач обирає необхідний тип даних	бажаний
Користувач	Вибір методу розв’язання	3. Користувач переходить до пункту обирання методу розв’язання у головному меню. 3.1. Система відображає список доступних методів розв’язання	обов'язковий
Користувач	Метод Зейделя	4. Користувач переходить до пункту обирання методу розв’язання у головному меню. 4.1. Система відображає список доступних методів розв’язання 4.2 Користувач обирає метод Зейделя	бажаний
Користувач	Метод простих ітерацій	5. Користувач переходить до пункту обирання методу розв’язання у головному меню. 5.1. Система відображає список доступних методів розв’язання 5.2 Користувач обирає метод Зейделя	обов'язковий

Продовження таблиці 4.1.

Користувач	Метод простих релаксацій	6. Користувач переходить до пункту обирання методу розв’язання у головному меню. 6.1. Система відображає список доступних методів розв’язання 6.2 Користувач обирає метод релаксацій	обов'язковий
Користувач	Введення початкових даних	7. Користувач переходить до пункту обирання початкових даних у головному меню. 7.1. Система відображає список пунктів підменю “задання початкових даних”	обов'язковий
Користувач	Задання параметрів	8. Користувач переходить до пункту обирання початкових даних у головному меню. 8.1. Система відображає список пунктів підменю “задання початкових даних” 8.2. Користувач обирає пункт “задання параметрів” 8.3. Система відображає список пунктів підменю “ задання параметрів ”	бажаний

Продовження таблиці 4.1.

Користувач	Кількість процесорних ядер	9. Користувач переходить до пункту обирання початкових даних у головному меню. 9.1. Система відображає список пунктів підменю “задання початкових даних” 9.2. Користувач обирає пункт “задання параметрів” 9.3. Система відображає список пунктів підменю “ задання параметрів ” 9.4. Користувач обирає пункт “кількість процесорних ядер” у підменю	бажаний
Користувач	Вибір файлу нейромережі	10. Користувач переходить до пункту обирання початкових даних у головному меню. 10.1. Система відображає список пунктів підменю “задання початкових даних” 10.2. Користувач обирає пункт “задання параметрів” 10.3. Система відображає список пунктів підменю “ задання параметрів ” 10.4. Користувач обирає пункт “вибір файлу нейромережі” у підменю	обов'язковий

Продовження таблиці 4.1.

Користувач	Вибір типу моделі	11. Користувач переходить до пункту обирання початкових даних у головному меню. 11.1. Система відображає список пунктів підменю “задання початкових даних” 11.2. Користувач обирає пункт “вибір типу моделі” 11.3. Система відображає список пунктів підменю “вибір типу моделі ”	бажаний
Користувач	Вибір меж розмірності	12. Користувач переходить до пункту обирання початкових даних у головному меню. 12.1. Система відображає список пунктів підменю “задання початкових даних” 12.2. Користувач обирає пункт “вибір типу моделі” 12.3. Система відображає список пунктів підменю “вибір типу моделі ” 12.4. Користувач обирає пункт “вибір меж розмірності” у підменю	бажаний
Користувач	Введення додаткової інформації	13. Користувач переходить до пункту обирання початкових даних у головному меню. 13.1. Система відображає список пунктів підменю “задання початкових даних” 13.2. Користувач обирає пункт “введення додаткової інформації” 13.3. Система відображає список пунктів підменю “ введення додаткової інформації ”	бажаний

Продовження таблиці 4.1.

Система	Вивід результатів	14. Система відображає результати в форматі обраним користувачем	обов'язковий
Система	Вивід у файл	15. Система відображає результати в форматі обраним користувачем 15.1. Вихідна інформація виводиться у файл	обов'язковий
Система	Консольний вивід	16. Система відображає результати в форматі обраним користувачем 16.1. Вихідна інформація виводиться в консоль	обов'язковий
Система	Аналіз моделі	17. Система аналізує структуру моделі	обов'язковий
Система	Визначення класу моделі	18. Система аналізує структуру моделі 18.1. Система відносить модель до певного класу	бажаний
Система	Визначення ступеню розрідженості	19. Система аналізує структуру моделі 19.1. Система відносить модель до певного класу 19.2. Система визначає ступінь розрідженості матриці	бажаний
Система	Класифікація рис	20. Система аналізує структуру моделі 20.1. Система відносить модель до певного класу 20.2. Система класифікує риси моделі	обов'язковий
Система	Обчислення моделі	21. Система обчислює модель обрану користувачем за обраним користувачем методом	обов'язковий

Продовження таблиці 4.1.

Система	Розбиття моделі	22. Система обчислює модель обрану користувачем за обраним користувачем методом 22.1. Система розбиває модель для обчислення	обов'язковий

4.3.1 Опис процесу діяльності

Розглянемо дії які періодично буде виконувати система при виборі користувачем введення даних, за допомогою UML діаграми діяльності (рисунок 4.2).

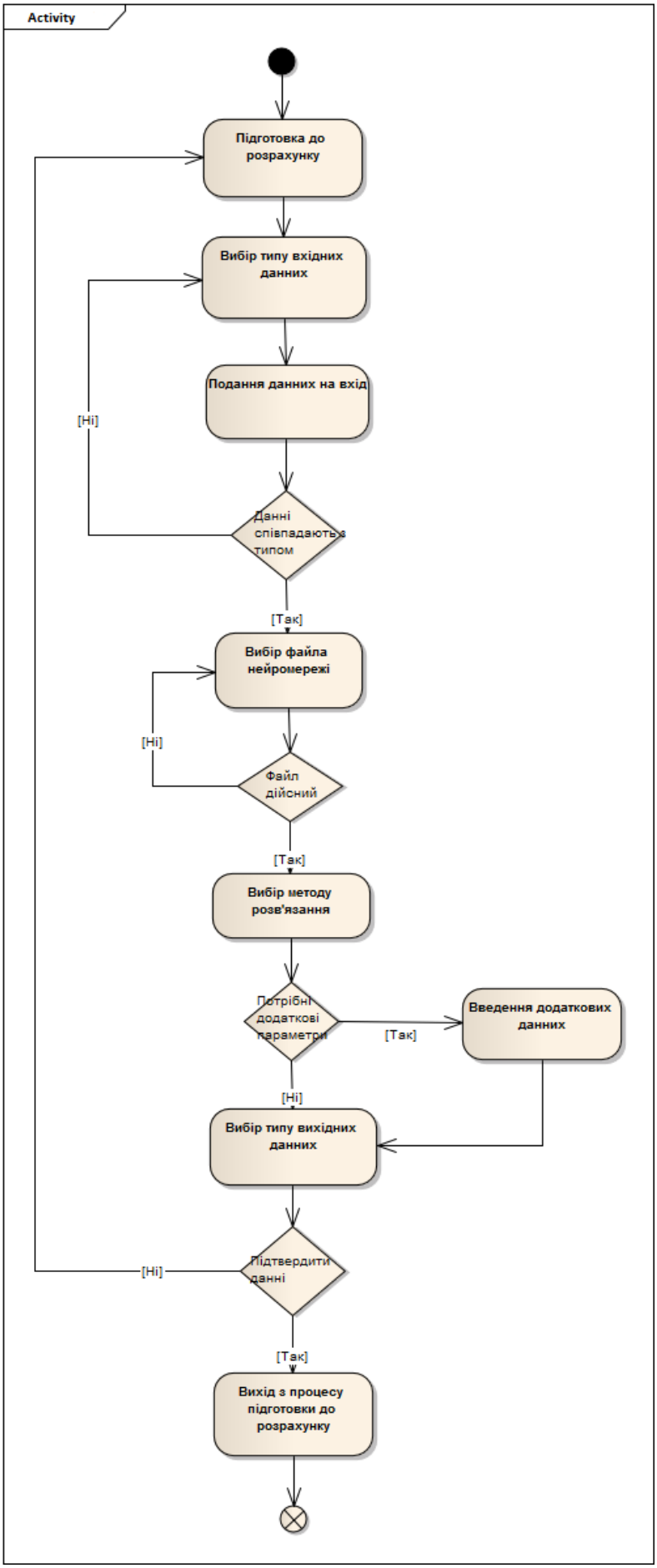

Рисунок 4.2 – Схема структурна діяльності. Процес введення даних для обробки

Детальний опис схеми структурної діяльності процесу внесення початкових даних наведено у таблиці 4.2.

Таблиця 4.2 – опис діяльності процесу внесення даних

Варіант діяльності	**Опис варіанту діяльності**
Підготовка до розрахунку	Загальна підготовка до введення даних (Головне меню)
Вибір типу вхідних даних	Обирається тип даних, який буде подаватися на вхід системи
Подання даних на вхід	Дані подаються на вхід до системи
Данні співпадають з типом	Перевірка на співпадіння даних які поданих на вхід із зарані обраним типом. Якщо, тип не співпадає – то повернення до вибору типу вхідних даних.
Вибір файла нейромережі	Файл нейромережі подається на вхід до системи
Файл дійсний	Перевірка на дійсність файлу поданного на вхід. Якщо, обраний файл не є дійсним – то повернення до вибору файлу нейромережі
Вибір методу розв’язання	Обирається метод, який буде використаний при подальшому розрахунку
Потрібні додаткові параметри	Обрання необхідності додання додаткових параметрів до даних наданих системі. Якщо, відповідь затверджена – то перехід до введення додаткових параметрів.
Введення додаткових параметрів	Додаткові параметри подаються на вхід до системи
Вибір типу вихідних даних	Обирається тип даних, який буде отриманий на виході із системи
Підтвердити дані	Перевірка на дійсність та коректність усієї поданої інформації. Якщо, інформація та дані незадовільні то перехід до підготовки до розрахунку

Продовження таблиці 4.2.

Вихід з процесу підготовки до розрахунку	Перехід до процесу аналізу та роботи с обраними даними.

4.3.2 Схема структурна класів

На рисунку 4.3 представлена структурна схема класів, які відповідають за виконання таких функцій програми, відображення інтерфейсу користувача, процесинг даних, їх аналіз і тд.

Діаграма містить 9 класів, а саме:

- «Data» – клас що виступає в ролі класу, для збереження даних та передачі їх від вводу користувача до інших класів.;
- «UI» – клас який представляє клас взаємодії з користувачем. А, саме прийняття введених даних, передача їх до класу «Data», відображення інформації, тощо;
- «Procesor» – клас який конвертує та перетворює інформацію до належного вигляду та інтерпретує результат аналізу задля подання її в клас «Solver»;
- «Solver» – клас який на підставі даних отриманих з класу «Procesor» займається вирішення систем рівнянь;
- «Model» – основний клас секції аналізу, працює з даними отриманими на вході та ініціалізує аналіз моделі;
- «Layer» – клас відповідальний за головну сторінку, сторінку розширеного перегляду та додання нових роздач;
- «Rating_count» – базовий клас усіх шарів нейромережі, який містить загальні методи;
- «Conv2D» – клас який представляє собою шар згортки. Є підкласом класу «Layer»;
- «MaxPooling» – клас який представляє собою шар агрегації (пулінгу). Є підкласом класу «Layer»;

– «Dense» – клас який представляє собою повноз’єднаний шар (fully-conected). Є підкласом класу «Layer»;

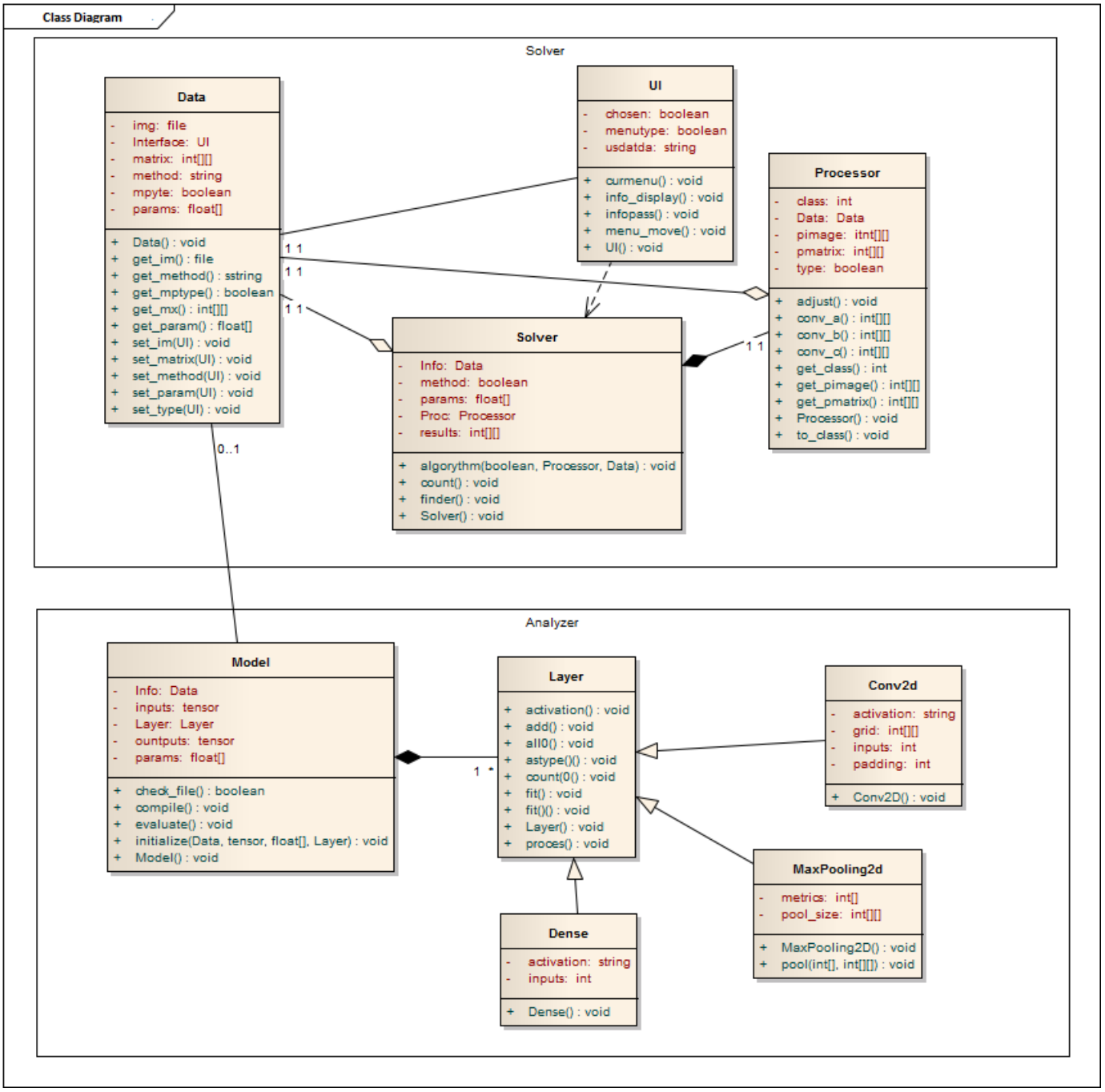

Рисунок 4.3 – Схема структурна класів

4.3.3 Діаграма послідовності

На рисунку 4.4 та 4.5 представлені схеми структурної послідовності. На даних семах представлені послідовності дій при передачі даних системою (рис.4.4) та при доданні користувачем інформації про файл нейромережі та ін

(рис 4.5). Також визначені актори та їх дії, що необхідні для виконання поставлених задач.

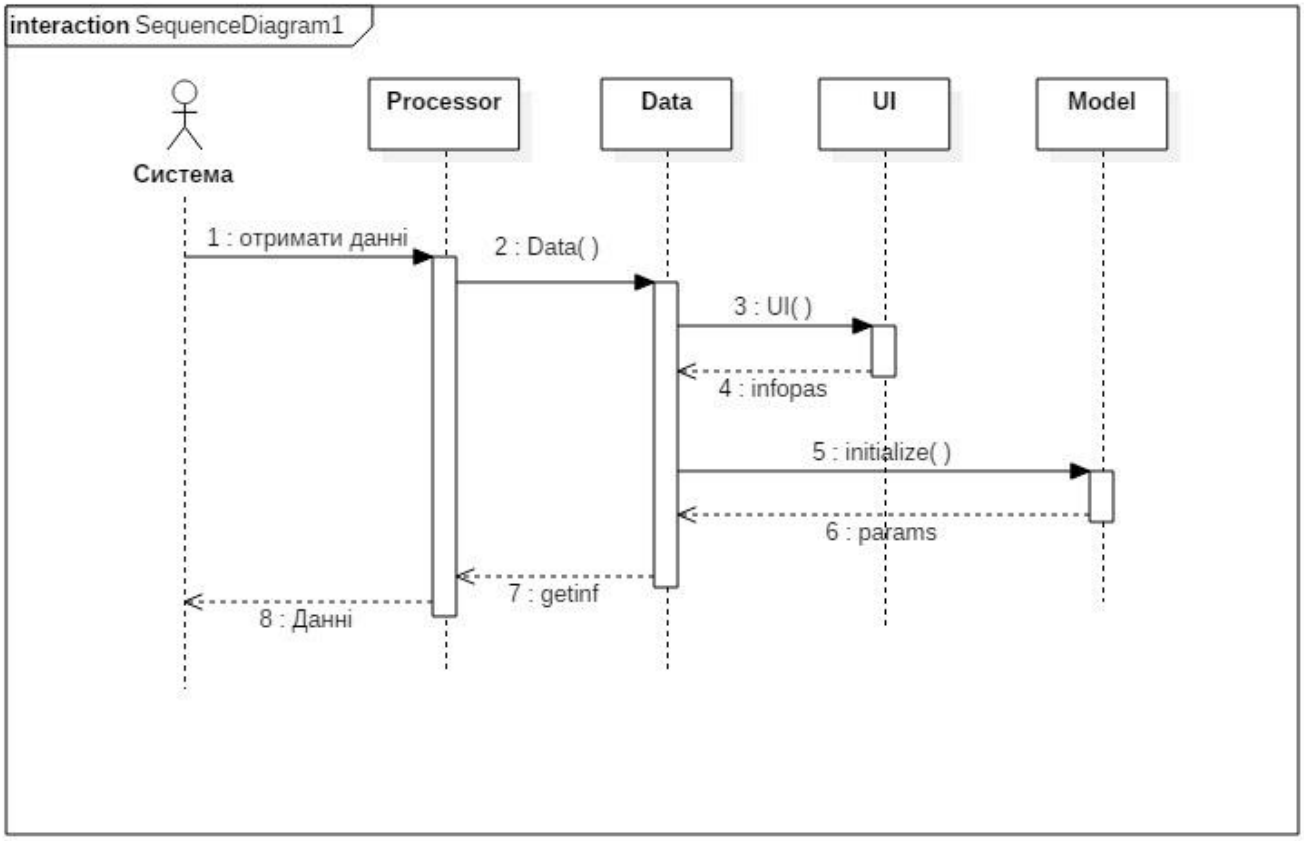

Рисунок 4.4 – Схема структурна послідовності передачі даних

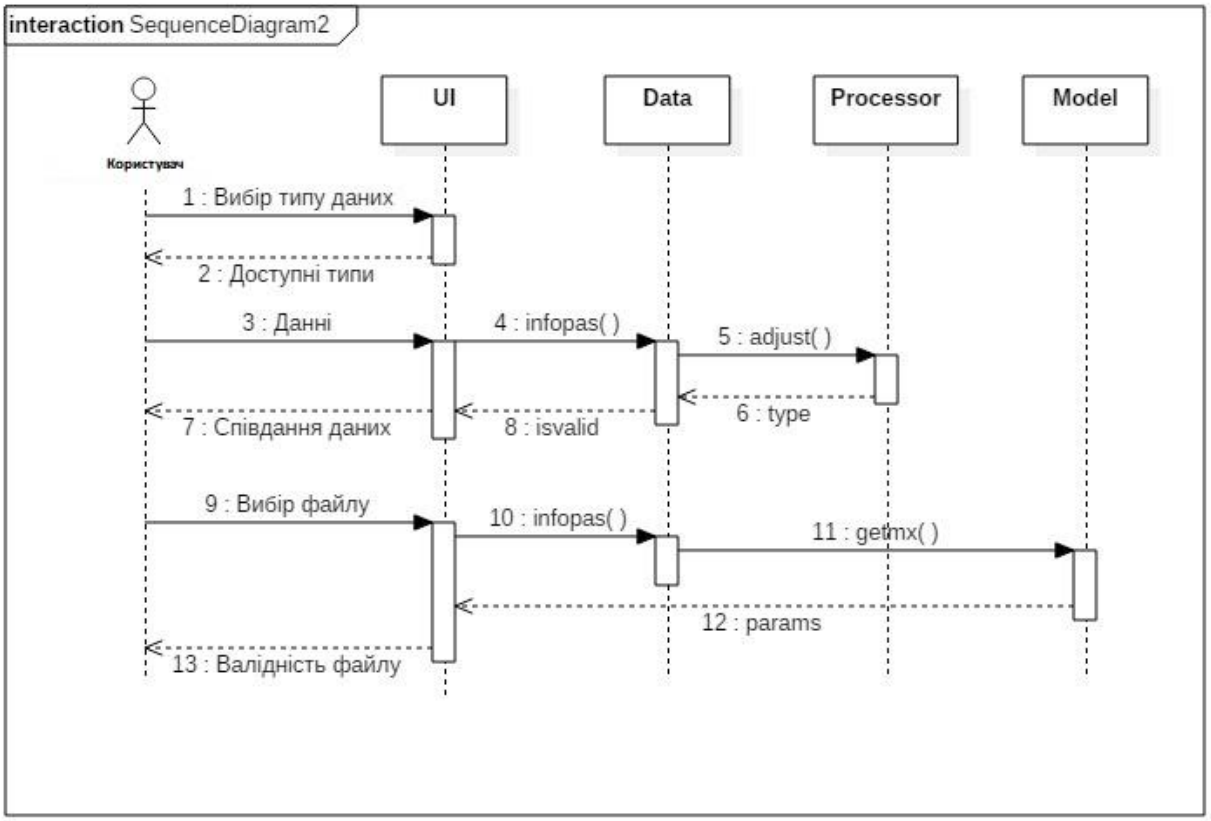

Рисунок 4.5 – Схема структурна послідовності додання даних користувачем

4.3.4 Діаграмма компонентів

На рисунку 4.6 представлена схема структурна компонентів, на якій зображено компоненти системи, що необхідні для функціонування ресурсу, та взаємозв'язки між ними.

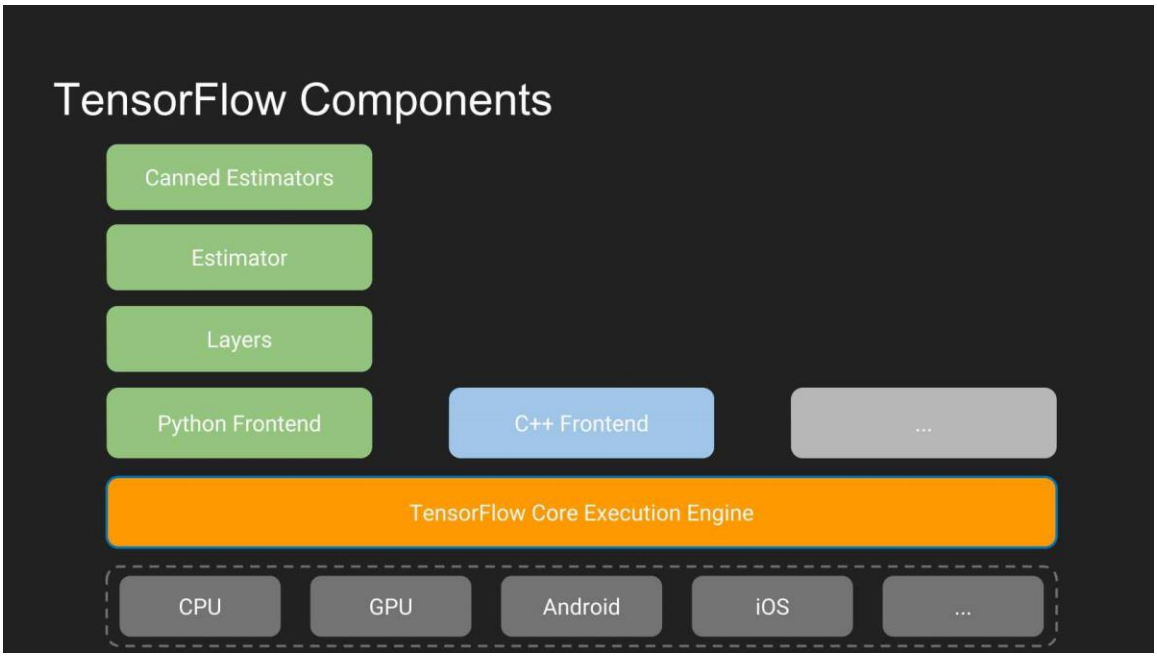

Рисунок 4.6 – Схема структурна компонентів

4.4 Висновок до розділу

Розглянуто основні особливості розробки програмних застосувань із нейромережами на мові програмування Python з використанням бібліотеки TensorFlow. До системи, що розробляється були поставлені мінімальні технічні вимоги для функціонування системи. Розроблена схема архітектури програмного забезпечення повністю відповідає поставленим вимогам. Під час розробки було сформовано та розроблено ієрархію класів, які відповідають поставленим вимогам та забезпечують необхідний функціонал системи. Описані особливості розробки системи, зображені діаграма класів, архітектура програмного забезпечення та мережа взаємодії системи.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Головне меню містить перелік наявних функцій для користувача та ступінь готовності до обрахунку. Головне меню показано на рисунку 5.1.

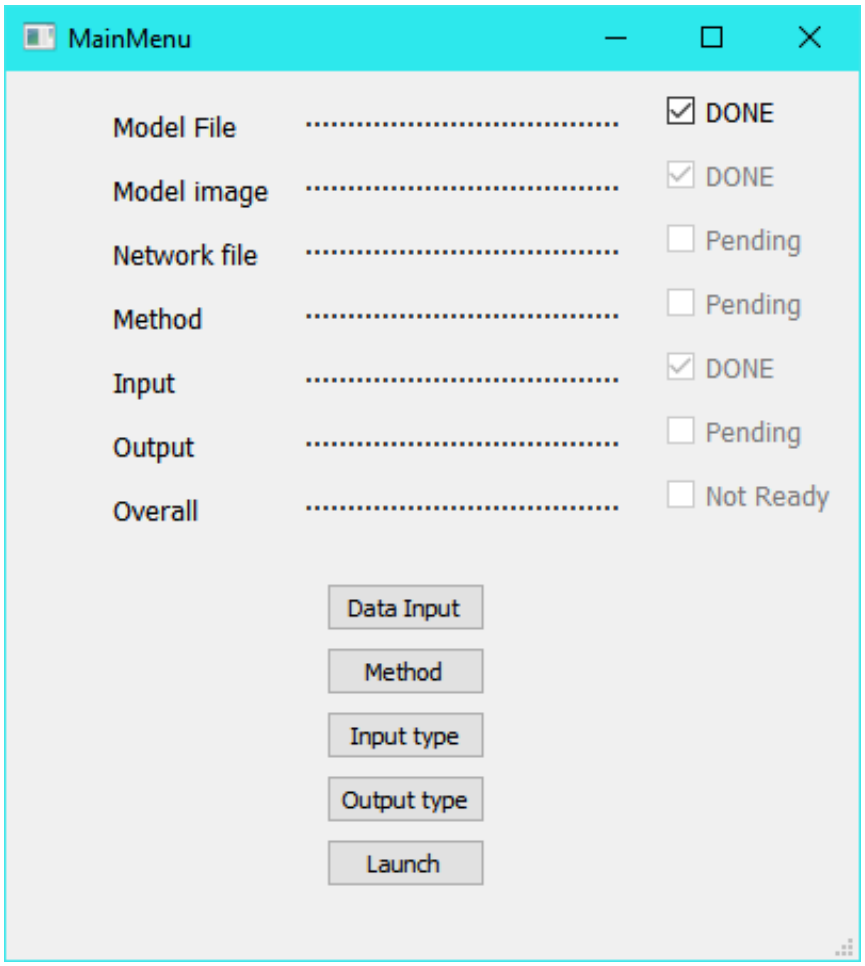

Рисунок 5.1 – Головне меню програми

Головне вікно програми містить відображення готовності до розрахунку та наступні вкладки:

- «Data input» – для вводу даних;
- «Method» – для вибору методу вирішення;
- «Input type» – для вибору типу вхідних файлів;
- «Output type» – для вибору типу вихідних файлів;
- «Launch» – для запуску та відображення результатів.

Для переходу до процесу введення даних необхідно перейти до «Data input». Опис вкладки «Data input» наведено на рисунку 5.2 1

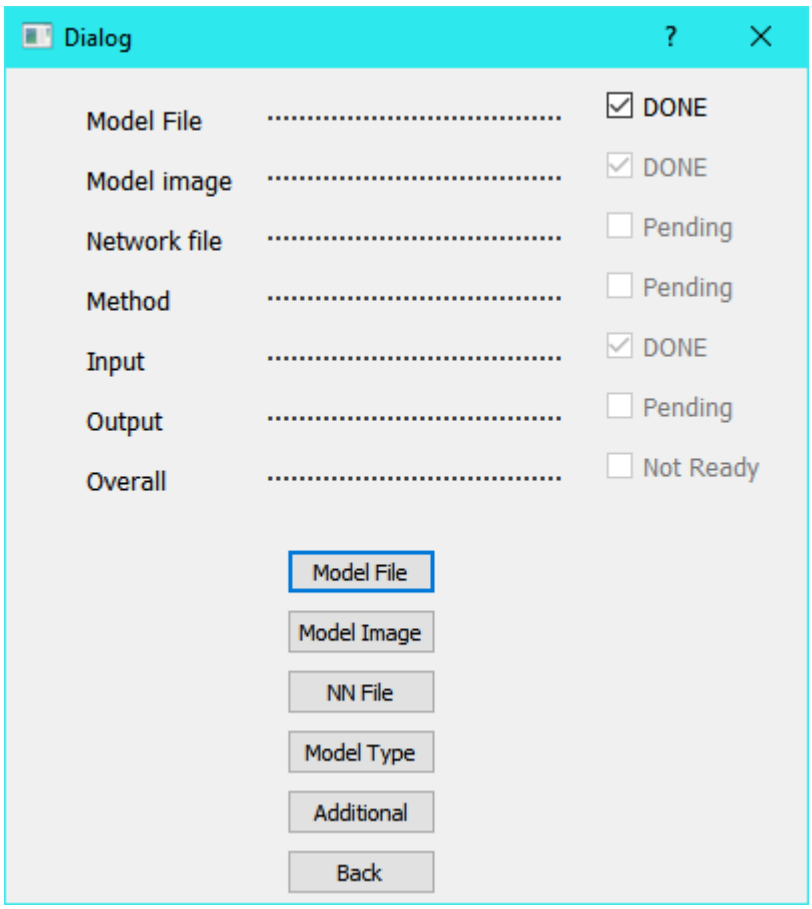

Рисунок 5.2 – Вкладка вводу даних

Вікно вводу даних програми містить відображення готовності до розрахунку та наступні вкладки:

- «Model File» – для вибору файла моделі;
- «Model Image» – для вибору зображення моделі;
- «NN File» – для вибору файла нейромережі;
- «Model Type» – для вибору типу моделі;
- «Additional» – для додаткової інформації;
- «Back» – для повернення до попереднього меню.

Для переходу до процесу додання файлу даних необхідно перейти до одного з пунктів мню «Model File», «Model Image» або «NN File». Опис вкладки

додання файлу наведено на рисунку 5.3

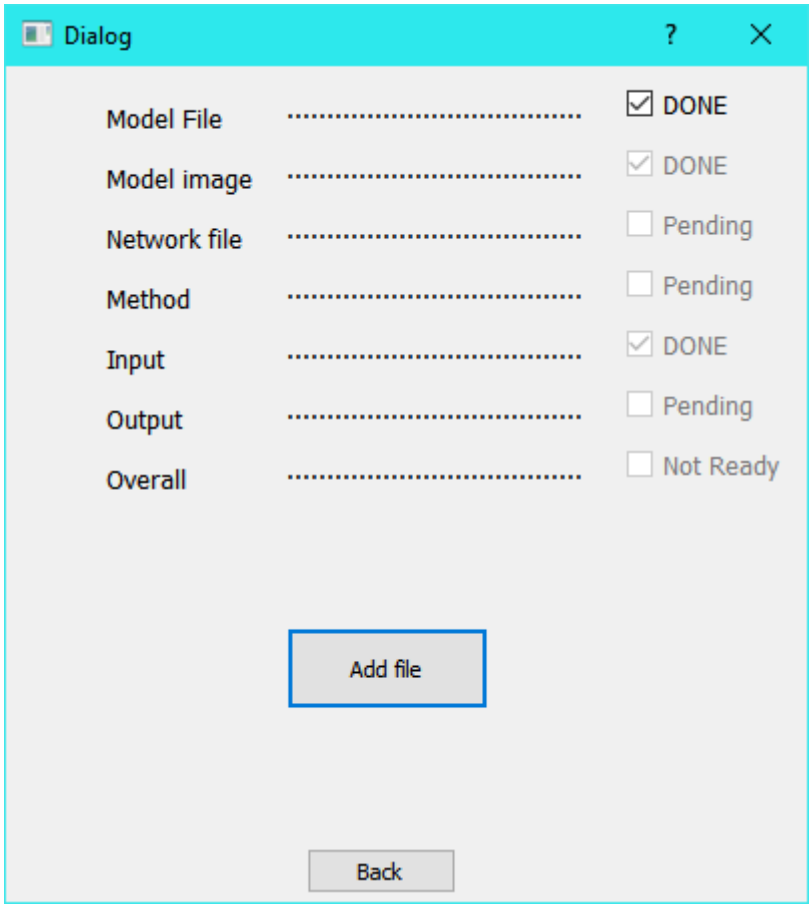

Рисунок 5.3 – Додання файлу

За допомогою кнопки «File add» можна завантажити обраний файл. Для переходу до процесу вибору методу необхідно з головного меню програми перейти до «Method». Опис вкладки «Method» наведено на рисунку 5.4.

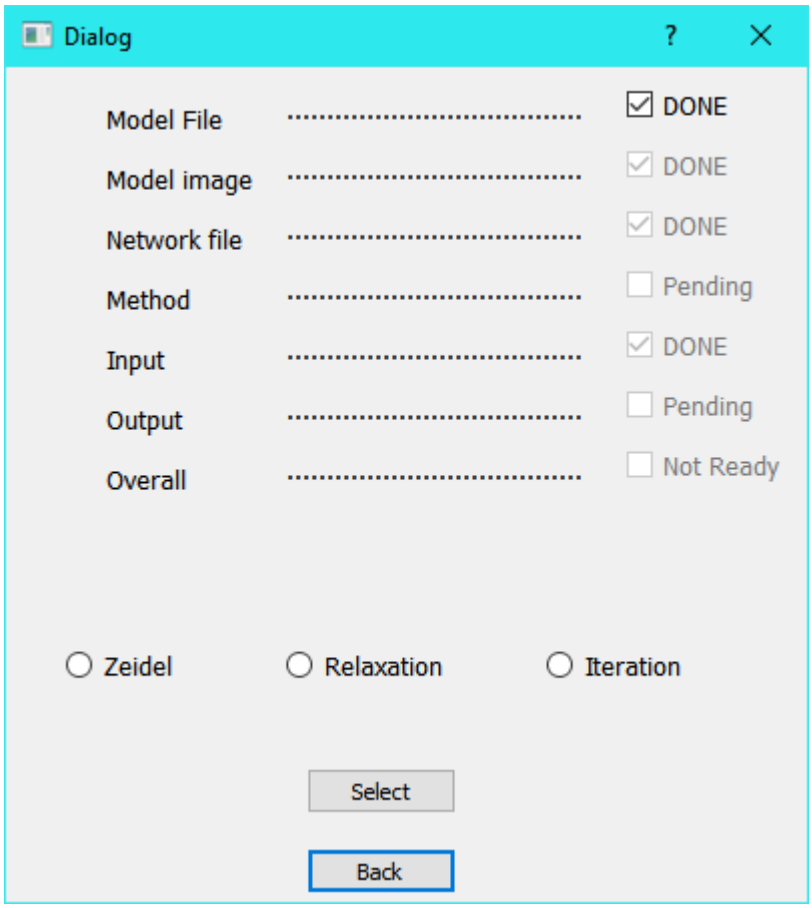

Рисунок 5.4 – Вибір методу розв’язання

За допомогою селекторів та кнопки «Select» можна обрати необхідний метод розв’язання. Для переходу до процесу вибору типу вхідного файлу необхідно з головного меню програми перейти до «Input». Опис вкладки «Input» наведено на рисунку 5.5.

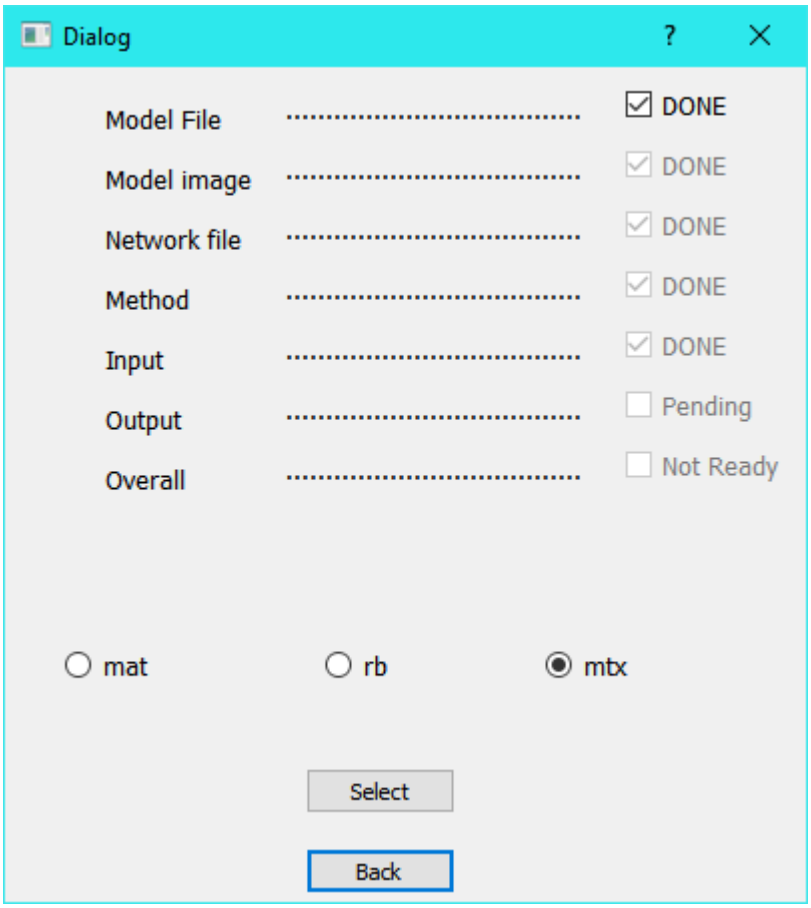

Рисунок 5.5 – Вибір типу вхідного файлу

За допомогою селекторів та кнопки «Select» можна обрати необхідний тип вхідного файлу. Для переходу до процесу вибору типу вихідних файлів необхідно з головного меню програми перейти до «Output». Опис вкладки «Output» наведено на рисунку 5.6.

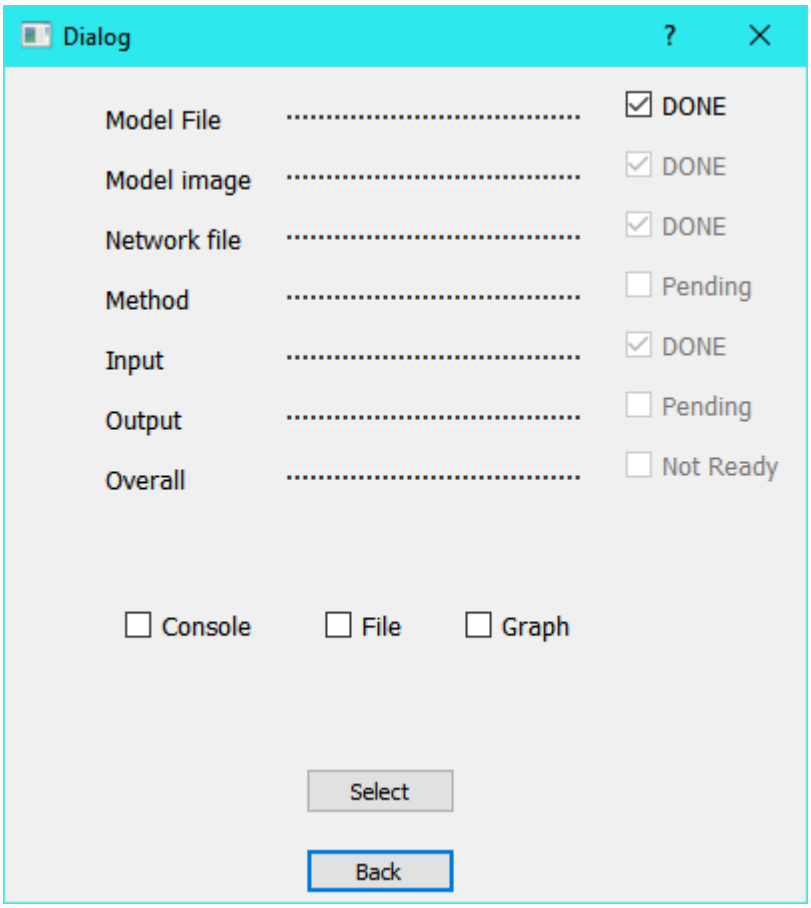

Рисунок 5.6 – Вибір типу вихідного файлу

Після заповнення усієї необхідної інформації можна перейти до вирішення задачі перейшовши до пункту «Launch» у головному меню програми. Після насиснення кнопки «Launch» з’явиться можливість підтвердтити раніше введену інформації або розпочати все спочатку. Вікно підтвердженя представлене на рисунку 5.7.

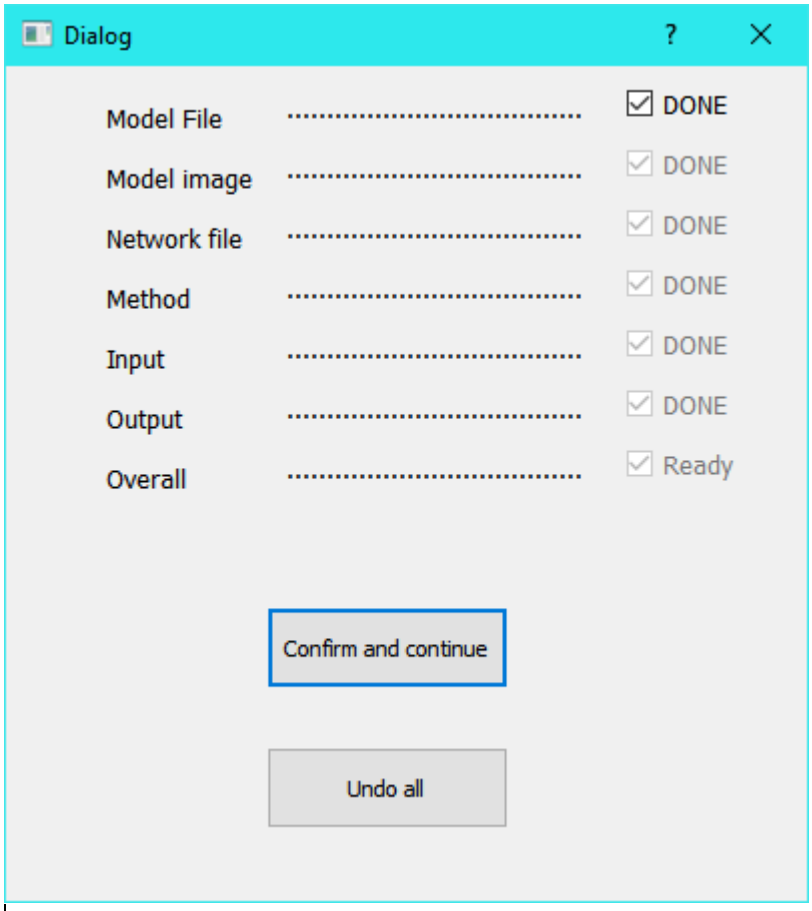

Рисунок 5.7 – Підтвердження правильності даних

5.2 Випробування програмного продукту

В цьому підрозділі наведено опис тестів і порядок їх виконання для перевірки відповідності ресурсу функціональним вимогам.

5.2.1 Мета випробувань

Метою випробувань є перевірка відповідності функцій паралельно адаптивного солвера для лінійних систем на основі нейромережі.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;

ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

В процесі тестування була перевірена функціональність системи за допомогою випробувань, що наведені у таблицях 5.1-5.15.

Таблиця 5.1 – Перехід до додання даних

Мета тесту:	**Перевірка функції «Перехід до додання даних»**
Початковий стан	Відкрито «Головне» меню програми
Вхідні данні:	
Схема проведення тесту	Перейти до пункту меню додання даних
Очікуваний результат:	Відображення меню додання даних
Стан після проведення випробувань:	Відкрите меню додання даних

Таблиця 5.2 – Додання файлу моделі

Мета тесту:	**Перевірка функції «Додання файлу моделі»**
Початковий стан	Відкрите меню додання даних
Вхідні данні:	Файл моделі заданого типу
Схема проведення тесту	Перейти у пункт меню “Model File”. Натиснути кнопку “Add file” та обрати необхідний файл
Очікуваний результат:	Обраний файл додався до системи. У головному меню відображається готовність файлу моделі.
Стан після проведення випробувань:	Відкрито «Головне» меню програми

Таблиця 5.3 – Додання зображення моделі

Мета тесту:	**Перевірка функції «Додання зображення моделі»**

Продовження таблиці 5.3.

Початковий стан	Відкрите меню додання даних
Вхідні данні:	Зображення моделі заданого типу
Схема проведення тесту	Перейти у пункт меню “Model File”. Натиснути кнопку “Add file” та обрати необхідний файл
Очікуваний результат:	Обране зображення додався до системи. У головному меню відображається готовність файлу моделі.
Стан після проведення випробувань:	Відкрито «Головне» меню програми

Таблиця 5.4 – Додання файлу нейромережі

Мета тесту:	**Перевірка функції «Додання файлу нейромережі»**
Початковий стан	Відкрите меню додання даних
Вхідні данні:	Файл нейромережі заданого типу
Схема проведення тесту	Перейти у пункт меню “Model File”. Натиснути кнопку “Add file” та обрати необхідний файл
Очікуваний результат:	Обраний файл додався до системи. У головному меню відображається готовність файлу моделі.
Стан після проведення випробувань:	Відкрито «Головне» меню програми

Таблиця 5.5 – Вибір типу виходу

Мета тесту:	**Перевірка функції «Вибір типу виходу»**
Початковий стан	Відкрите меню додання даних
Вхідні данні:	Вибір проміж можливих варіантів виводу.
Схема проведення тесту	Обрати один або декілька варіантів виходу та натиснути кнопку “Select”
Очікуваний результат:	Обраний тип додався до системи. У головному меню відображається готовність файлу моделі.

Продовження таблиці 5.5.

Стан після проведення випробувань:	Відкрито «Головне» меню програми

Таблиця 5.6 — Вибір типу входу

Мета тесту:	**Перевірка функції «Вибір типу входу»**
Початковий стан	Відкрите меню додання даних
Вхідні данні:	Вибір проміж можливих варіантів вводу.
Схема проведення тесту	Обрати один варіант виходу та натиснути кнопку “Select”
Очікуваний результат:	Обраний тип додався до системи. У головному меню відображається готовність файлу моделі.
Стан після проведення випробувань:	Відкрито «Головне» меню програми

Таблиця 5.7 – Вибір методу розв’язання

Мета тесту:	**Перевірка функції «Вибір методу розв’язання»**
Початковий стан	Відкрито «Головне» меню програми
Вхідні данні:	Вибір проміж можливих методів
Схема проведення тесту	Користувач натискає кнопку “Method”. Обрати один варіант виходу та натиснути кнопку “Select”
Очікуваний результат:	Обраний метод додався до системи. У головному меню відображається готовність файлу моделі.
Стан після проведення випробувань:	Відкрито «Головне» меню програми

Таблиця 5.8 – Вибір типу моделі

Мета тесту:	**Перевірка функції «Вибір типу моделі»**
Початковий стан	Відкрите меню додання даних
Вхідні данні:	Обраний тип

Продовження таблиці 5.8.

Схема проведення тесту	Користувач натискає кнопку “Model type”. Після чого обрає один варіант виходу та натискає кнопку “Select”
Очікуваний результат:	Обраний тип додався до системи. У головному меню відображається готовність типу моделі.
Стан після проведення випробувань:	Відкрито «Головне» меню програми

Таблиця 5.9 – Додання додаткової інформації

Мета тесту:	**Перевірка функції «Додання додаткової інформації»**
Початковий стан	Відкрите меню додання даних
Вхідні данні:	Додаткова інформація
Схема проведення тесту	Користувач натискає кнопку “Additional”. Після чого обрає вводить необхідні дані та натискає кнопку “Select”
Очікуваний результат:	Обрана інформація додалась до системи.
Стан після проведення випробувань:	Відкрито «Головне» меню програми

Таблиця 5.10 – Перевірка на заповненні дані

Мета тесту:	**Перевірка функції «Перевірка на заповненні дані»**
Початковий стан	Відкрита «Головна» сторінка сайту
Вхідні данні:	Вся інформація вибрана чи додана користувачем. Але не в повному обсязі.
Схема проведення тесту	Користувач натискає кнопку “Launch”.
Очікуваний результат:	Система перевіряє усі надані дані.
Стан після проведення випробувань:	Відкрита «Головна» сторінка сайту

Таблиця 5.11 Перевірка на заповненні дані

Мета тесту:	**Перевірка функції «Перевірка на заповненні дані»**
Початковий стан	Відкрита «Головна» сторінка сайту
Вхідні данні:	Вся інформація вибрана чи додана користувачем.В повному обсязі.
Схема проведення тесту	Користувач натискає кнопку “Launch”.
Очікуваний результат:	Система перевіряє усі надані дані.
Стан після проведення випробувань:	Система відображає результати розрахунку

Таблиця 5.12 – Скидання даних

Мета тесту:	**Перевірка функції «Скидання даних»**
Початковий стан	Відкрита сторінка підтвердження готовності до розрахунку
Вхідні данні:	Вся інформація вибрана чи додана користувачем
Схема проведення тесту	Користувач натискає кнопку “Undo all”.
Очікуваний результат:	Система видаляє усі дані.
Стан після проведення випробувань:	Відкрито «Головне» меню програми

5.3 Тестування точності нейромережі

Розроблена нейромережа була протестована на тесових даних відібраних з ресурсу Matrix Market. Результати перевірки точності роботи нейромережі можна побачити на рисунку 5.8.

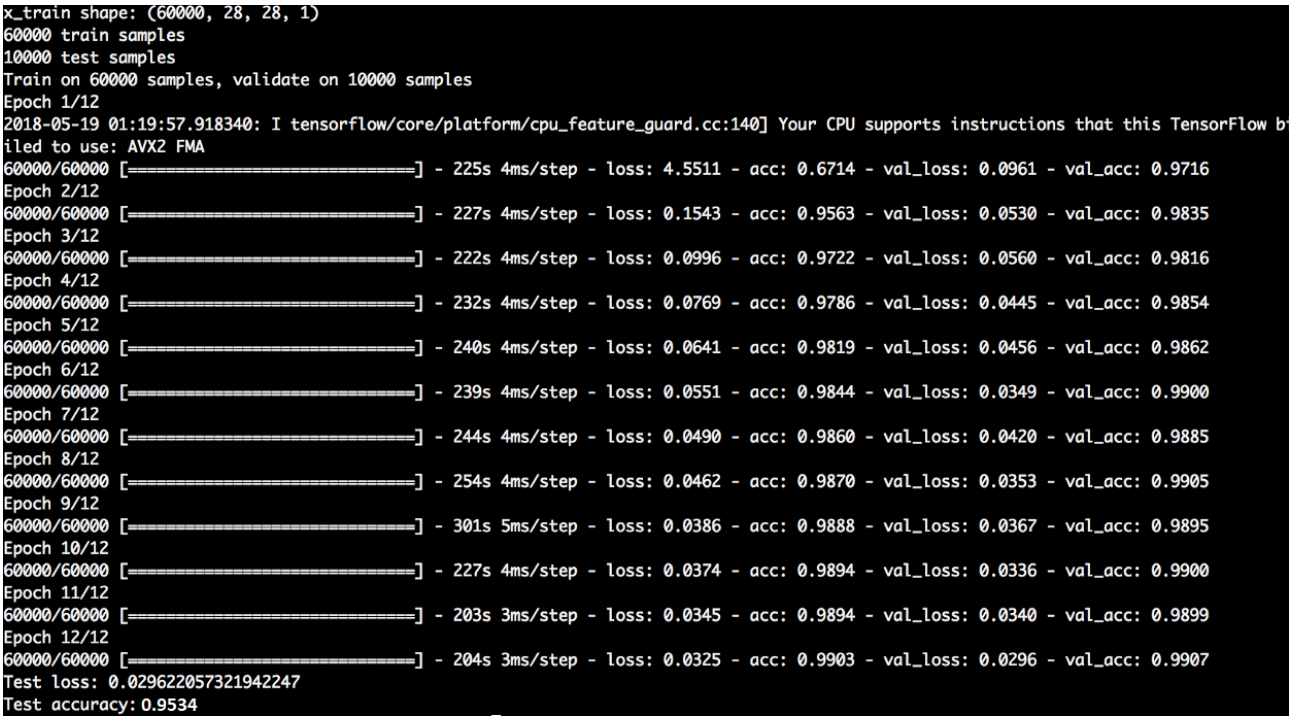

Рисунок 5.8 – Тестування точності нейромережі

Як можно побачити з рисунку 5.8 у результаті тренування була досягнута точність приблизно 96%, що є непоганим резльтатом але потребує дообробки.

5.4 Висновок до розділу

В даному розділі було описано детальне керівництво користувача та наведені приклади екранних форм програми. Керівництво користувача включає опис дій, які можуть виконуватись користувачем. Представлені основні сценарії для тестування програми щодо відповідності функціональним вимогам.

ВИСНОВКИ

У даній магістерській дисертації було створено програму для вирішення слар з розрідженими матрицями на основі нейромережі. Програмна реалізація системи написана мовою програмування Python на платформі PyQt на базі операційної системи Windows. Такий вибір зроблено на користь використання розвиненого середовища розробки Qt та легкої можливості портування програми на будь яку іншу платформу, зокрема варто відмітити наявність зручних статистичних функцій та функцій створення різноманітних графіків і інших методів, що дозволяють задовольнити вимогу створення зручного графічного інтерфейсу користувача. Першим етапом розробки цього проекту був аналіз предметної області. Були досліджені останні теоретичні роботи у даній сфері, виокремлено та використано в роботі результати найбільш перспективних із них. На основі проведених досліджень стало можливим зробити обґрунтований вибір найбільш ефективного для даної задачі алгоритму і оптимальних значень його настроюваних параметрів. Була розроблена нейромережа для аналізу ступіню розрідженості у матриціх. В рамках опису архітектури побудованої системи розглянуто перелік усіх її модулів, наведено діаграми компонентів, що можуть бути утворені в результаті динамічного зв'язування компонент. У якості документації дана інструкція користувача з ілюстраціями. Таким чином в результаті проведеної роботи була досягнута поставлена мета дослідження.

ПЕРЕЛІК ПОСИЛАНЬ

1. **Джордж А.** Численное решение больших разреженных систем уравнений / Джордж А., Дж. Лиу. – М.: Мир, 1984. – 390 с.
2. Джордж А., Лиу Дж. Численное решение больших разреженных систем уравнений. – М.: Мир, 1984. – 333 с.
3. Дж. Голуб, Ч. Ван Лоун Матричные вычисления: Пер. с англ. – М.: Мир, 1999.
4. Р. Тюарсон, Разреженные матрицы: Пер. с англ. – М.: Мир, 1977.-192.с
5. Блатов И.А., Пименов А.С., Бубнова Н.В. Псевдоразреженные матрицы и прикладной вейвлет анализ //Системы управ. и информ. Технологий. Научно – техн. журнал. – Москва – Воронеж. – 2006.– № 1(23). – С. 68 – 73. [9] Блатов И.А., Рогова Н.В. Ме
6. Компьютерное моделирование: моделирование как метод научного познания [Електронний ресурс] // Режим доступу: http://www.econf.rae.ru/article/6722
7. Компьютерное моделирование [Електронний ресурс] // Режим доступу: http://www.inf1.info/book/export/html/215
8. Области применения компьютерного моделирования [Електронний ресурс] // Режим доступу: http://bourabai.ru/cm/6.htm
9. Баландин М.Ю. Методы решения СЛАУ большой размерности / М.Ю.Баландин, Э.П.Шурина – Новосибирск: изд-во НГТУ, 2000. – 70с.
10. Блатов И.А. Методы решения систем с разреженными матрицами / Блатов И.А., Глушакова Т.Н., М.Е. Эскаревская – Воронеж: Воронежский гос. ун-т, 2002. – 34 с.
11. Брамеллер А. Слабозаполненные матрицы / Брамеллер А., Аллан Р., Хэмэм Я. – М.: Энергия, 1979. – 192 с.

12. Глушакова Т.Н. Методы работы с разреженными матрицами произвольного типа / Глушакова Т.Н. Эксаревская М.Е. – Воронеж: Воронежский гос. ун-т, 2005. – 44 с.
13. Глушакова Т.Н. Методы решения систем с разреженными матрицами / Глушакова Т.Н., Блатова И.А. – Воронеж: Воронежский гос. ун-т, 2000. – 36 с.
14. Годунов С.К. Решение систем линейных уравнений / Годунов С.К. – Новосибирск: Наука, 1980.
15. Недашковський М.О. Обчислення з λ-матрицями / Недашковський М.О., Ковальчук О.Я. – К.: Наук. думка, 2007. – 294 с.
16. Saad Y. Iterative Methods for Sparse Linear Systems / Saad Y. – PWS Publishing Company, 2000, 448p.
17. Хіміч О.М. Гібридний алгоритм розв'язування лінійних систем зі стрічковими матрицями прямими методами // О.М. Хіміч, А.Ю Баранов // Комп'ютерна математика : Зб. наук. праць. – 2013. – Т. 2. – С. 80–87.
18. Писанецки С. Технология разреженных матриц / Писанецки С. – М.: Мир, 1988. — 410 с.
19. Эстербю О. Прямые методы для разреженных матриц / О. Эстербю, З. Златев – М.: Мир, 1987. – 118 с.
20. Ильин В.П. Экспериментальный анализ явных методов неполной факторизации / Ильин В.П., Ицкович Е.А. // Сборник научных трудов «Численные методы и математическое моделирование» под ред. Ильина В.П. – Новосибирск: ВЦ СО АН СССР – 1990, с. 85-94.
21. SUPERLU, a sequential library for the direct solution of large, sparse, nonsymmetric systems of linear equations on high performance machines [Електронний ресурс] / Sherry Li, Jim Demmel, John Gilbert, Laura Grigori. – 2008. – Режим доступу : http://crd.lbl.gov
22. Naumov M. Parallel Incomplete-LU and Cholesky Factorization in the Preconditioned Iterative Methods on the GPU. – 19 p. – [Електронний

ресурс] Режим доступу: https://research.nvidia.com/sites/default/files/publications/nvr-2012-003.pdf

23. Перельмутер А.В., Фиалко С.Ю.Прямые и итерационные методы решения большеразмерных конечно-элементных задач строительной механики // Киев, Киевский национальный технический университет строительства и архитектуры - 2009, 6с.
24. Сушко Г.Б. Многопоточная параллельная реализация итерационного алгоритма решения систем линейных уравнений с динамическим распределением нагрузки по нитям вычислений / Г.Б. Сушко, С.А. Харченко – 2009, 6с.
25. Городецкий А.С. Компьютерные модели конструкции / Городецкий А.С., Евзеров И.Д. – К.: Факт, 2005. – 334 с.
26. Михалевич В.С. Численные методы для многопроцессорного вычислительного комплекса ЕС / Михалевич В.С., Бик Н.А., Брусникин Б.Н.,…, Химич А.Н. и др./ Под редакцией И.Н. Молчанова.– М.: Издание ВВИА им. проф. Н.Е. Жуковского, 1986. – 401 с.
27. Молчанов И.Н. Некоторые методы решения больших разреженных систем уравнений на многопроцессорном вычислительном комплексе (МВК) / И.Н. Молчанов, И.В. Решетуха, О.В. Рудич, С.П. Семенченко. – К., 1991. – 31 с. – (Препр. / АН УССР. Ин-т кибернетики им. В.М. Глушкова; 91-2).
28. Intel® Math Kernel Library (Intel® MKL) – Режим доступу: https://software.intel.com/en-us/intel-mkl
29. Garbow B.S. Matrix Eigensystem Routine / Garbow B.S., Royle J.M., Dongarra J.J., Moller M.M. – EISPACK Guide Extension Lecture Notes in Computer Science, vol. 51. Springer-Verlag, 193. – 236 p
30. Химич А.Н. Численное программное обеспечение MIMD– компьютера Инпарком / Химич А.Н., Молчанов И.Н.,… Полянко В.В., и др. – Киев: Наукова думка, – 2007. – 222 с.

31. Chow E. ILUS: an incomplete LU Preconditioner in Sparse Skyline format / Chow E., Saad Y. – In: Int. J. for Num. methods.
32. Saad Y. ILUT: a dual threshold incomplete ILU factorization / Saad Y. // Report umsi-92-38, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, 1992, - 18 p.
33. Alfredo Buttari, Julien Langou, Jakub Kurzak, and Jack Dongarra: A Class of Parallel Tiled Linear Algebra Algorithms for Multicore Architectures.Parallel Computing, Volume 35, Issue 1, P. 38-53, 2009, ISSN:0167-8191.
34. Капорин И.Е. Постфильтрация множителей IC2-разложения для балансировки параллельного предобусловливания / Капорин И.Е., Коньшин И.Н. // Журнал вычислительной математики и математической физики. – 2009. – Т. 49, № 6. – С.940-957.
35. Молчанов И.Н. О реализации методов преобусловливания на много процессорных системах / Молчанов И.Н., Рябцев В.Е. // Оптимизация численных методов решения задач на ЭВМ : сб. науч. тр. – К.: Институт Кибернетики им. В.М. Глушкова АН УСРС, 1986. – С. 44–48.
36. Мулярчик С.Г. Численное моделирование микроэлектронных структур Мулярчик С.Г. – Минск: Университетское, 1989. – 368 с.
37. Гладкий А. В. Моделирование переноса загрязнений в атмосфере с использованием параллельных вычислений / А. В. Гладкий, В. А. Богаенко // Управляющие системы и машины. - 2014. - № 6. - С. 18–26.
38. Statistical lists of supercomputers [Електронний ресурс] / Hans Meuer, Erich Strohmaier, Jack Dongarra, Horst Simon. – Waibstadt-Daisbach. – 2009. – Режим доступу : http://www.top500.org
39. Фаддеева В.Н. Параллельные вычисления в линейной алгебре / Фаддеева В.Н., Фаддеев Д.К. // Кибернетика.- 1982.- № 3.- С. 13 - 31.
40. Timothy A. Davis, Direct Methods for Sparse Linear Systems, SIAM, Philadelphia, Sept. 2006.

41. Freeman T.L. Parallel numerical algorithms / Freeman T.L., Phillips C. – New York: Prentice Hall International, 1992.- 315 p.
42. Geist A. PVM 3 User's Guide and Reference Manual / Geist A., Beguelin A., Dongarra J., Jiang W., Mancheck R., Sunderam V. – Tennessee: – Energy Systems, Inc.- 1993. - 78 p.
43. Golub G. Scientific computing. An introduction with parallel computing / Golub G., Ortega J.M. – Boston: Academic Press, 1993.- 442 p.
44. Молчанов И.Н. Машинные методы прикладных задач. Алгебра, приближение функций / Молчанов И.Н. – Киев: Наукова думка. 1987. – 288 с.
45. Воеводин В.В. Математические модели и методы в параллельных процессах / Воеводин В.В. – М.: Наука, 1986.- 256 с.
46. Воеводин В.В. Ошибки округлений и устойчивость в прямых методах линейной алгебры / Воеводин В.В. – М.: ВЦ МГУ, 1969. – 153 с.
47. Quinn M.J. Parallel Computing. Theory and practice / Quinn M.J. – New York:- 1994. – 446 p.
48. Нестеренко Б.Б. Основы асинхронных методов параллельных вычислений / Нестеренко Б.Б., Марчук В.А. – Киев: Наук. думка, 1989. – 176 с.
49. Молчанов И.Н. Об одном пакете программ для решения систем линейных алгебраических уравнений / Молчанов И.Н, Николенко Л.Д., Кириченко М.П. // Кибернетика. – 1972, № 1. – С. 127 – 134.
50. Ортега Дж. Введение в параллельные и векторные методы решения линейных систем / Ортега Дж. – М.: Мир, 1991.
51. Букатов А. А. Программирование многопроцессорных вычислительных систем / Букатов А. А., Дацюк В. Н., Жегуло А. И. – Ростов-на-Дону. Издательство ООО «ЦВВР», 2003. – 208 с.
52. Технологии параллельного программирования [Електронний ресурс] // Лаборатория Параллельных Информационных Технологий, НИВЦ МГУ. – 2009. – Режим доступу : http://parallel.ru/

53. W. Gropp Using MPI-2: Advanced Features of the Message-Passing Interface / W. Gropp, E. Lusk, and R. Thakur. – Cambridge: MIT Press, 1999. – 382 p.
54. Naumov M. Preconditioned Block-Iterative Methods on GPUs // Special Issue: 83rd Annual Meeting of the International Association of Applied Mathematics and Mechanics (GAMM). – 2012 – Volume 12, Issue 1. – P. 11–14.
55. Фиалко С.Ю. Применение современных вычислительных технологий к расчету многоэтажных зданий //Киев, Институт механики им. С.П.Тимошенко НАН Украины – 2009, 5с.
56. Капорин И.Е. Параллельное решение симметричных положительно-определенных систем на основе перекрывающегося разбиения на блоки / Капорин И.Е. Коньшин И.Н. // Журнал вычислительной математики и математической физики. – 2001. – Т. 41, № 4. – С.515-528.
57. Богачев К. Ю. Метод капорина-коньшина параллельной реализации блочных предобусловливателей для несимметричных матриц в задачах фильтрации многокомпонентной смеси в пористой среде / К. Ю. Богачев, Я. В. Жабицкий // Вестник Московского университета. Серия 1. Математика. Механика, № 1, 2010, С. 46-52.
58. Naumov M. Parallel Solution of Sparse Triangular Linear Systems in the Preconditioned Iterative Methods on the GPU. – 21 p. – [Електронний ресурс] Режим доступу: https://research.nvidia.com/sites/default/files/publications/nvr-2011-001.pdf
59. PSparslib, A Portable Library of Parallel Sparse Iterative Solvers [Електронний ресурс] / Yousef Saad, Gen-Ching Lo, Sergey Keznetsov. – 1998. – Режим доступу : http://citeseerx.ist.psu.edu
60. PSBLAS: Parallel Sparse Basic Linear Algebra Subroutines [Електронний ресурс] / Salvatore Filippone, Alfredo Buttari. – 2008. – Режим доступу : http://claudius.ce.uniroma2.it

61. MUMPS, a MUltifrontal Massively Parallel sparse direct Solver [Електронний ресурс] / Iain DUFF. – 2008. – Режим доступу : http://www.cerfacs.fr
62. Банч Дж.. Развитие математического программного обеспечения / Банч Дж.., Воеводин В.В. // Численный анализ на фортране. – Сборник научных трудов, вып. 17. – М.: МГУ. – 1976.- С. 3 – 8
63. Глушков В.М. О наборе программ для решения систем линейных алгебраических уравнений на мащинах серии МИР / Глушков В.М., Молчанов И.Н., Николенко Л.Д. // Кибернетика. – 1968, № 6. – С. 1 – 6.
64. Глушков В.М. Программное обеспечение ЭВМ МИР-1 и МИР-2. Численные методы / В.М. Глушков, И.Н. Молчанов, Б.Н. Брусникин и др.- Киев: Наук. думка, 1976. – Т. 1. – 280 с
65. Райс Дж. Матричные вычисления и математическое обеспечение / Райс Дж. – М.: Мир, 1984. – 264 с.
66. Форсайт Дж. Численное решение систем линейных алгебраических уравнений / Форсайт Дж., Моулер К. – М.: Мир, 1969. – 153 с.
67. Уилкинсон Дж.Х. Справочник алгоритмов на языке АЛГОЛ. Линейная алгебра / Уилкинсон Дж.Х., Райнш К. – М.: Машиностроение, 1976. – 389 с.
68. Разработка программ вычислительного типа. – М.: Изд-во Моск. ун-та, - 1990. – 124 с.
69. Jack Dongarra. Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition / Jack Dongarra – 1995
70. A Catalog of Control Data Software. Control Data Corporation – Minnesota, 1974. – 420 p.
71. Nag. Fortran Library. Introductory Guide. – Oksford, 1992. – 15. – 105 p.
72. Aztec: A Massively Parallel Iterative Solver Library for Solving Sparse Linear Systems [Електронний ресурс] / Ray S. Tuminaro, John N. Shadid, Mike Heroux. – 1999. – Режим доступу : http://www.cs.sandia.gov

73. BlockSolver95: Scalable Library Software for the Parallel Solution of Sparse Linear Systems [Електронний ресурс] / Mark T. Jones, Paul E. Plassmann. 1997. – Режим доступу : http://www-unix.mcs.anl.gov
74. IBM System/360 Scientific Subroutine Package. Version 11. Programmer's Manual. – New York, 1967. – 420 p.
75. Moler C.B. Matlab user's guide / Moler C.B. // Univ. New Mexico. – 1981. – P. 1 - 6.
76. Newbery A.C.R. The Boing library and handbook of mathematical routines / Newbery A.C.R. // Mathematical Software – New York, 191. – 420 p.
77. PSPASES, Parallel SPArse Symmetric dirEct Solver [Електронний ресурс] / Anshul Gupta, Mahesh Joshi, George Karypis, Vipin Kumar, Fred Gustavson. – 1999. – Режим доступу : www.cs.umn.edu
78. PARALUTION – The library for iterative Sparse Methods on CPU and GPU [Електронний ресурс] // Режим доступу: http://www.paralution.com/
79. Saad Y. SPARSKIT: a basik tool kit for sparse matrix computations / Saad Y. – 1994.
80. Тимошенко С.П. Теория упругости / Тимошенко С.П., Гудьер Дж. – М.: Наука, 192. – 52 с.
81. О.Ю. Грищенко Розпаралелювання різницевих схем на основі ДС-алгоритму / О.Ю. Грищенко, А.С. Марцафей, В.С. Федорова. // Доповіді Академії Наук України. – 2011. – 7 – С. 78-83.
82. Документація по Python. [Електронний ресурс] – Режим доступу до ресурсу : https://www.python.org/doc/
83. Документація по TensorFlow. [Електронний ресурс] – Режим доступу до ресурсу : https://www.tensorflow.org/programmers_guide/

ДОДАТОК А

ГРАФІЧНИЙ МАТЕРІАЛ

ПЛАКАТ 1 СХЕМА СТРУКТУРНА ВАРІАНТІВ ВИКОРИСТАННЯ

ПЛАКАТ 2 СХЕМА СТРУКТУРНА КЛАСІВ

ПЛАКАТ 3 СХЕМА СТРУКТУРНА ДІЯЛЬНОСТІ

ПЛАКАТ 4 СХЕМА СТРУКТУРНА ПОСЛІДОВНОСТІ

ПЛАКАТ 5 СХЕМА СТРУКТУРНА ПОСЛІДОВНОСТІ

ПЛАКАТ 6 СТРУКТУРА НЕЙРОМЕРЕЖІ

ПЛАКАТ 7 ЕКРАННІ ФОРМИ